

Databases

02 Data modeling

Dr. Lucas Iacono

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Human-Centred Computing

Course calendar

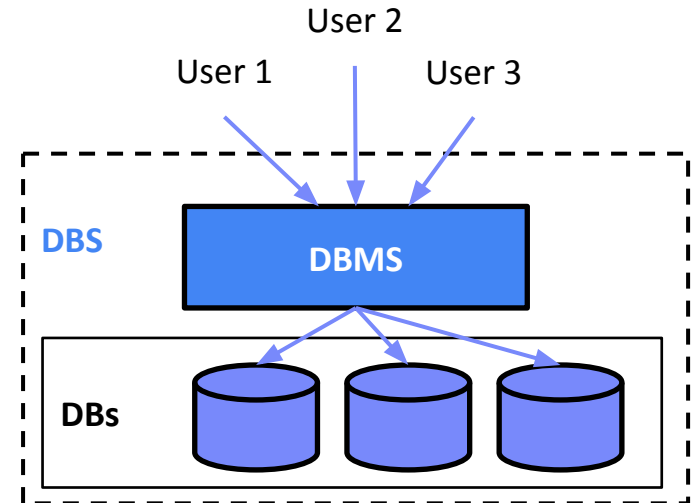
- Part 1: Data Modeling and relational databases
 - (1) Mon. 20.10.2025: Overview of databases and data storage systems + **Group Project**
 - **(2) Mon. 27.10.2025: Data modeling + Exercise 1**
 - (3) Mon. 03.11.2025: Relational databases and normalization (+ bit of SQL for Ex. 1)
 - (4) Mon. 10.11.2025: Query languages (SQL) + **Exercise 2 presentation**

Last week

- Introduction to databases
- Course organization
- Group project

Last week

- Introduction to databases
- Course organization
- Group project
- What is a database system?
 - Database system (DBS): DBMS + DBs
 - DBMS: Database Management System (Software to handle DBs)
 - DBs: Database (data/metadata collection to describe something)
 - Note: DB also a short for DBS/DBMS



This week: Agenda

- Database design
- Entity-relationship (ER) models and diagrams

This week: Agenda

- Database design
- Entity-relationship (ER) models and diagrams

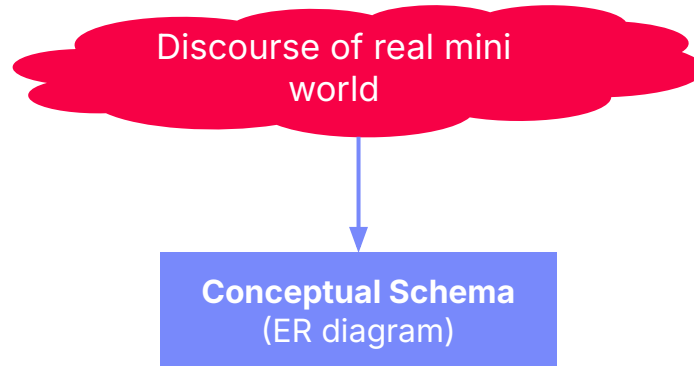
- Exercise 1: data modeling

Database design

Data Modeling

- Concept for describing data objects and their relationships
- **Schema:** Description (structure, semantics) of specific data collection

This lecture



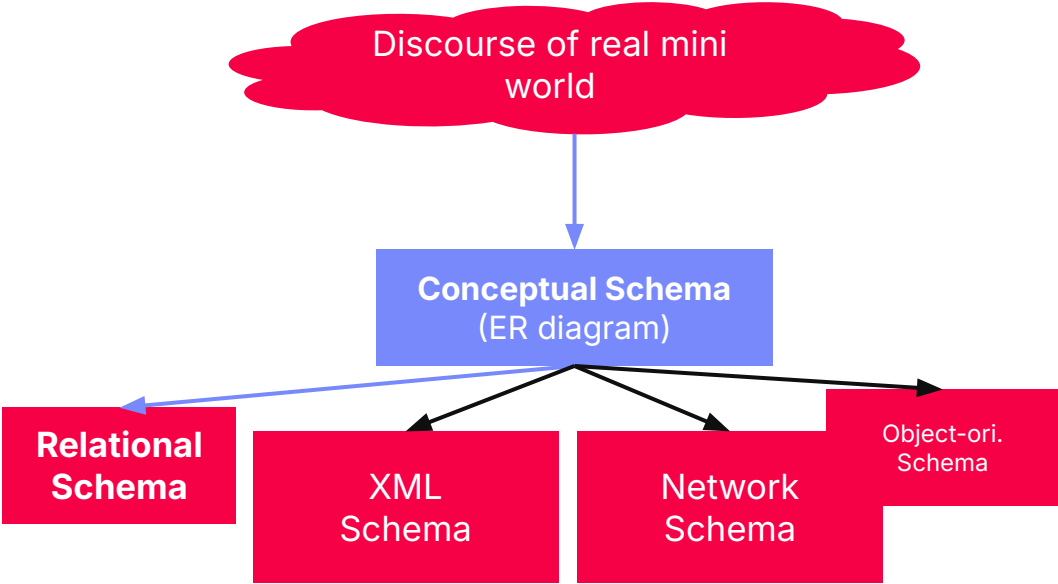
Manual Modeling

Data Modeling

- Concept for describing data objects and their relationships
- **Schema**: Description (structure, semantics) of specific data collection

This lecture

Next lecture



Manual Modeling

Semi-automatic Transformation

Data Models

- Conceptual Data Models
 - **Entity-Relationship Model (ERM)**, focus on data, ~1975
 - Unified Modeling Language (UML), focus on data and behavior (e.g., object-oriented class diagram), ~1990

Data Models

- Conceptual Data Models
 - **Entity-Relationship Model (ERM)**, focus on data, ~1975
 - Unified Modeling Language (UML), focus on data and behavior (e.g., object-oriented class diagram), ~1990

- Logical Data Models
 - **Relational data model**
 - Key-value
 - Document (XML, JSON)
 - Graph
 - Time series
 - ...

DB Design Lifecycle Phases

#1 Requirements engineering

- Collect and analyze data and application requirements
- **Specification documents**

DB Design Lifecycle Phases

#1 Requirements engineering

- Collect and analyze data and application requirements
- Specification documents

#2 **Conceptual Design** (this lecture, exercise 1 and project)

- **Model data semantics and structure, independent of logical data model**
- **ER model / diagram**

DB Design Lifecycle Phases

#1 Requirements engineering

- Collect and analyze data and application requirements
- Specification documents

#2 Conceptual Design (this lecture, exercise 1 and project)

- Model data semantics and structure, independent of logical data model
- ER model / diagram

#3 Logical Design (next lecture, exercise 1 and project)

- Model data with implementation details of concrete data model
- e.g., relational schema + integrity constraints, permissions, etc

DB Design Lifecycle Phases

#1 Requirements engineering

- Collect and analyze data and application requirements
- Specification documents

#2 Conceptual Design (this lecture, exercise 1 and project)

- Model data semantics and structure, independent of logical data model
- ER model / diagram

#3 Logical Design (next lecture, exercise 1 and project)

- Model data with implementation details of concrete data model
- e.g., relational schema + integrity constraints, permissions, etc

#4 Physical Design (not covered in this course, see „Data Management“ if interested)

- Model **user-level data organization** in a specific DBMS (and data model)
- Account for **deployment environment** and **performance requirements**

Tool Support

- #1 **Visual Design Tools**
 - Draw ER diagrams in any presentation software (e.g., MS PowerPoint, LibreOffice)
 - Many desktop or web-based tools support ER diagrams directly (e.g., MS Visio, creately.com)
- #2 **Design Tools w/ Code Generation**
 - Draw and validate ER diagrams
 - Generate relational schemas as SQL data definition language scripts
 - **Examples:** SAP (Sybase) PowerDesigner, MS Visual Studio plugins (SQL server), etc.

Note: For the exercise and project, please use the basic drawing tool **draw.io**

Entity-relationship (ER) models and diagrams

Chen Notation



[Peter P. Chen: The Entity-Relationship Model - Toward a Unified View of Data. **ACM Trans. Database Syst.** 1(1) 1976]

[Peter P. Chen: The Entity-Relationship Model: Toward a Unified View of Data. **VLDB** 1975]



ER Diagram Components (Chen Notation)

- **Entity (noun)**
 - Entities are objects of the real world
 - An entity instance is a concrete instantiation of an entity
 - Rectangular shape + capitalized



Employee

Example: Entity Instance

| | | |
|-----------------|--|---|
| Herr | lacono , Lucas Emanuel, PhD |  |
| E-Mail | liacono(at)tugraz.at | |
| Homepage | https://lucasiacono.github.io/ | |
| Telefon TU Graz | - | |
| Mobil | 4331687332824 | |
| Sprechstunde | 09:00 - 17:00 | |
| Zusatzinfo | In case you require consultation hours please coordinate in advance via e-mail | |
| Postadresse | 7060 Institute of Human-Centred Computing 8010 Graz, Sandgasse 36/III | |

Forschung & Lehre

-  Abschlussarbeiten
-  Lehrveranstaltungen
-  Forschungsportal (PURE)

ER Diagram Components (Chen Notation)

- **Entity (noun)**
 - Entities are objects of the real world
 - An entity instance is a concrete instantiation of an entity
 - Rectangular shape + capitalized

- **Relationship (verb)**
 - Relationships are concrete associations of entities
 - Diamond shape + **non-capitalized**



ER Diagram Components (Chen Notation)

- **Entity (noun)**
 - Entities are objects of the real world
 - An entity instance is a concrete instantiation of an entity
 - Rectangular shape + capitalized

- **Relationship (verb)**
 - Relationships are concrete associations of entities
 - Diamond shape + non-capitalized

- **Attribute**
 - Entities or relationships are characterized by attribute-value pairs
 - **Round shape + capitalized**
 - Multi-valued attributes (e.g., phone numbers)



Multi-valued attributes



ER Diagram Components (Chen Notation)

- **Keys**

- Attributes that **uniquely identify** an entity
- **Every entity** type **must have** such a **key**
- Natural or surrogate (artificial) keys

EmpID

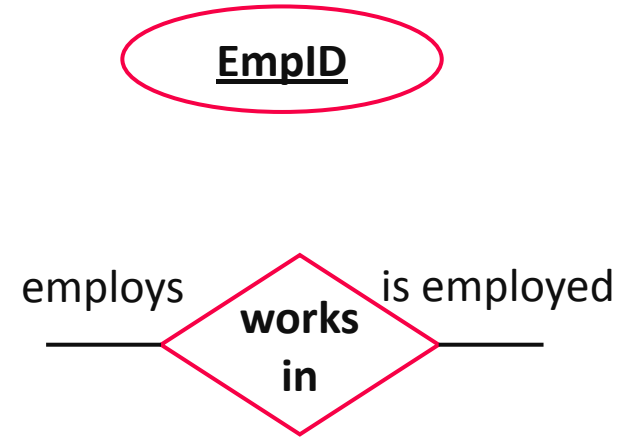
ER Diagram Components (Chen Notation)

- **Keys**

- Attributes that uniquely identify an entity
- Every entity type must have such a key
- Natural or surrogate (artificial) keys

- **Role**

- Optional description of relationship types
- Useful for recursive relationships
- E.g., an employee is employed in a project but a project employs an employee



An EmployeeDB Example

Department

Employee

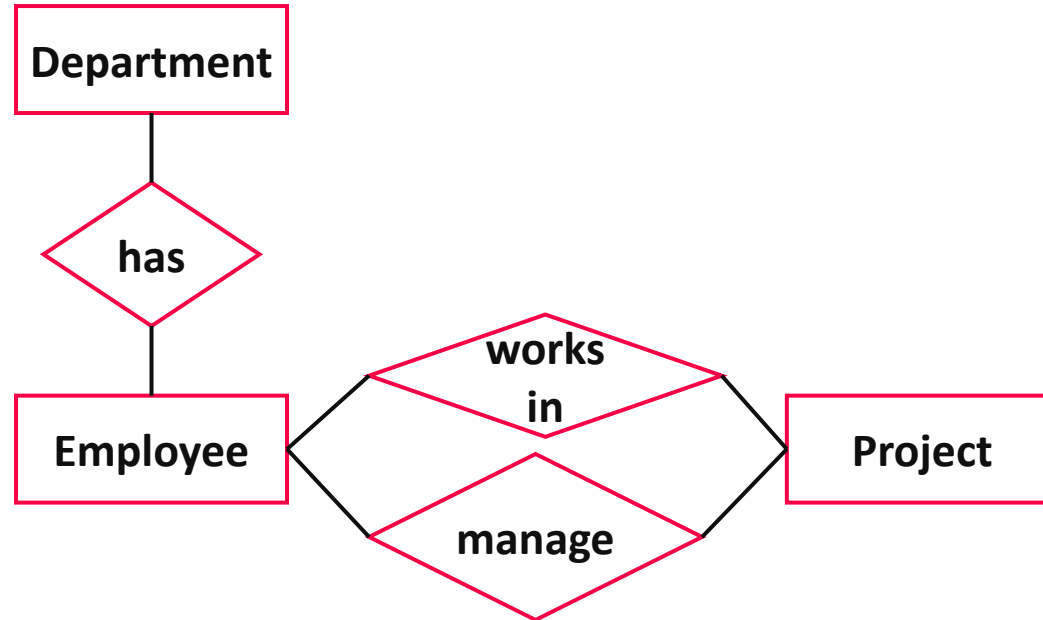
Project

[Peter P. Chen: The Entity-Relationship Model - Toward a Unified View of Data.
ACM Trans. Database Syst. 1(1) 1976]



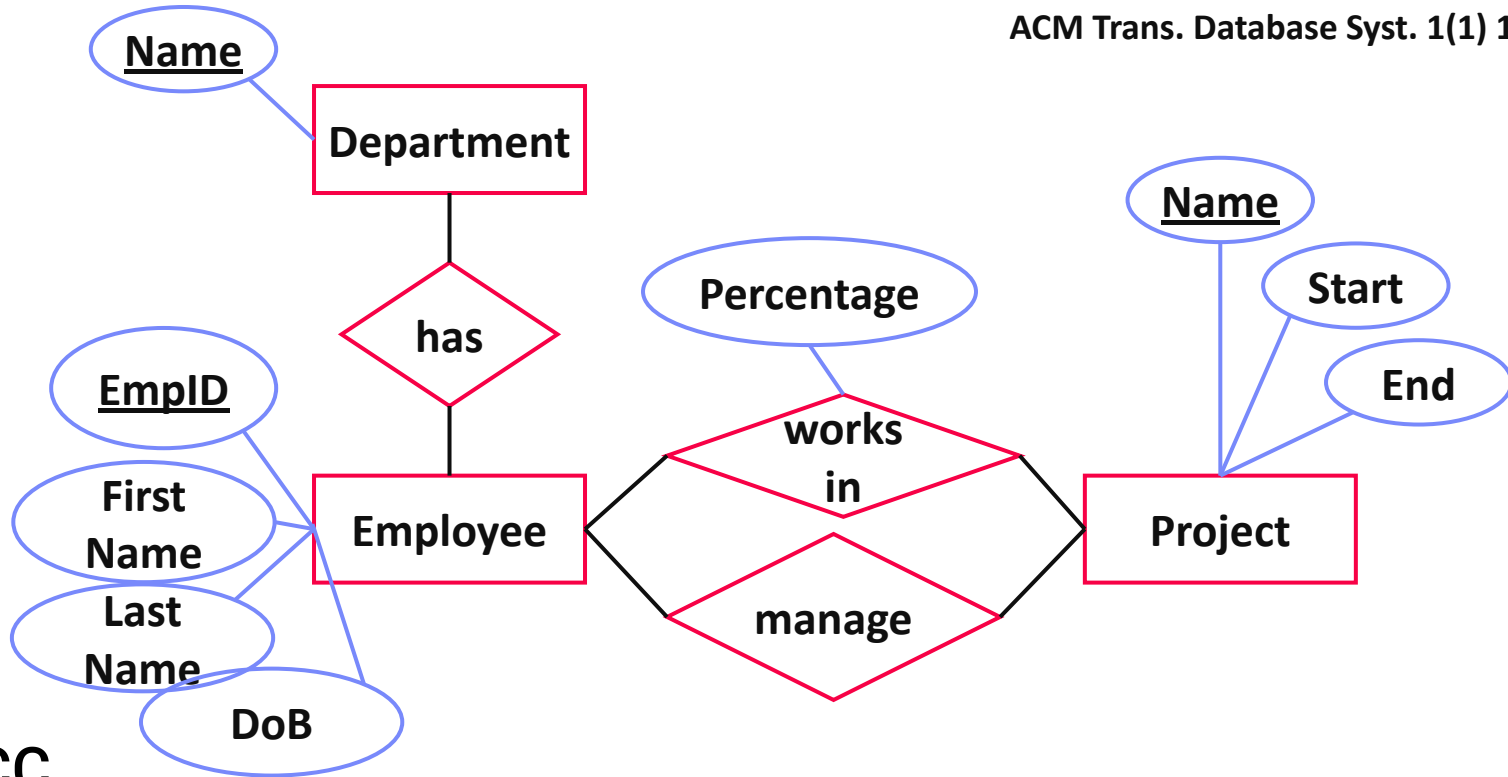
An EmployeeDB Example

[Peter P. Chen: The Entity-Relationship Model - Toward a Unified View of Data. *ACM Trans. Database Syst.* 1(1) 1976]

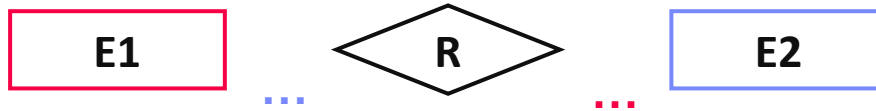


An EmployeeDB Example

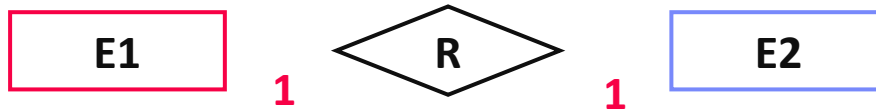
[Peter P. Chen: The Entity-Relationship Model - Toward a Unified View of Data. ACM Trans. Database Syst. 1(1) 1976]



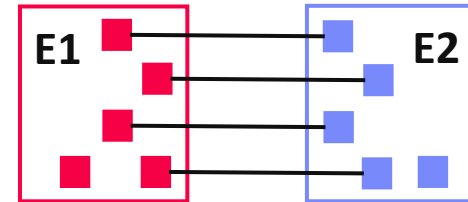
Multiplicity/Cardinality in Chen Notation



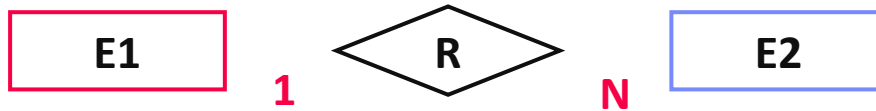
Multiplicity/Cardinality in Chen Notation



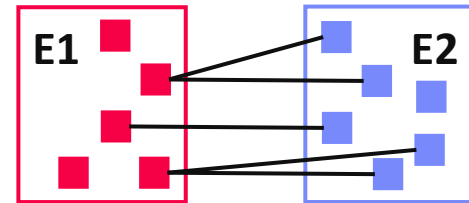
- 1:1 (one-to-one)
 - Each e1 relates to at most one e2
 - Each e2 relates to at most one e1
 - E.g., 1 employee can manage 1 department and 1 department can be managed by 1 employee



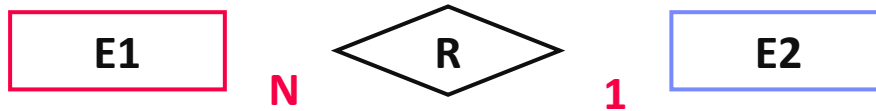
Multiplicity/Cardinality in Chen Notation



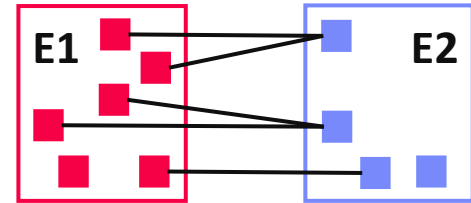
- 1:N (one-to-many)
 - Each e1 relates to many e2 (0,1,...N)
 - Each e2 relates to at most one e1
 - E.g., 1 department can have N employees but 1 employee can only have 1 department



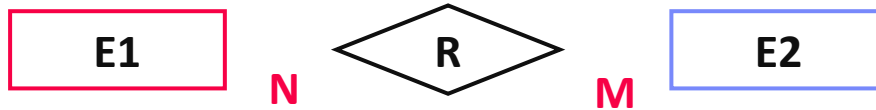
Multiplicity/Cardinality in Chen Notation



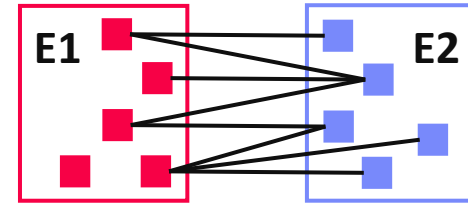
- N:1 (many-to-one)
 - Each e2 relates to many e1 (0,1,...N)
 - Each e1 relates to at most one e2
 - E.g., 1 employee can manage N projects but 1 project can be managed by just 1 employee



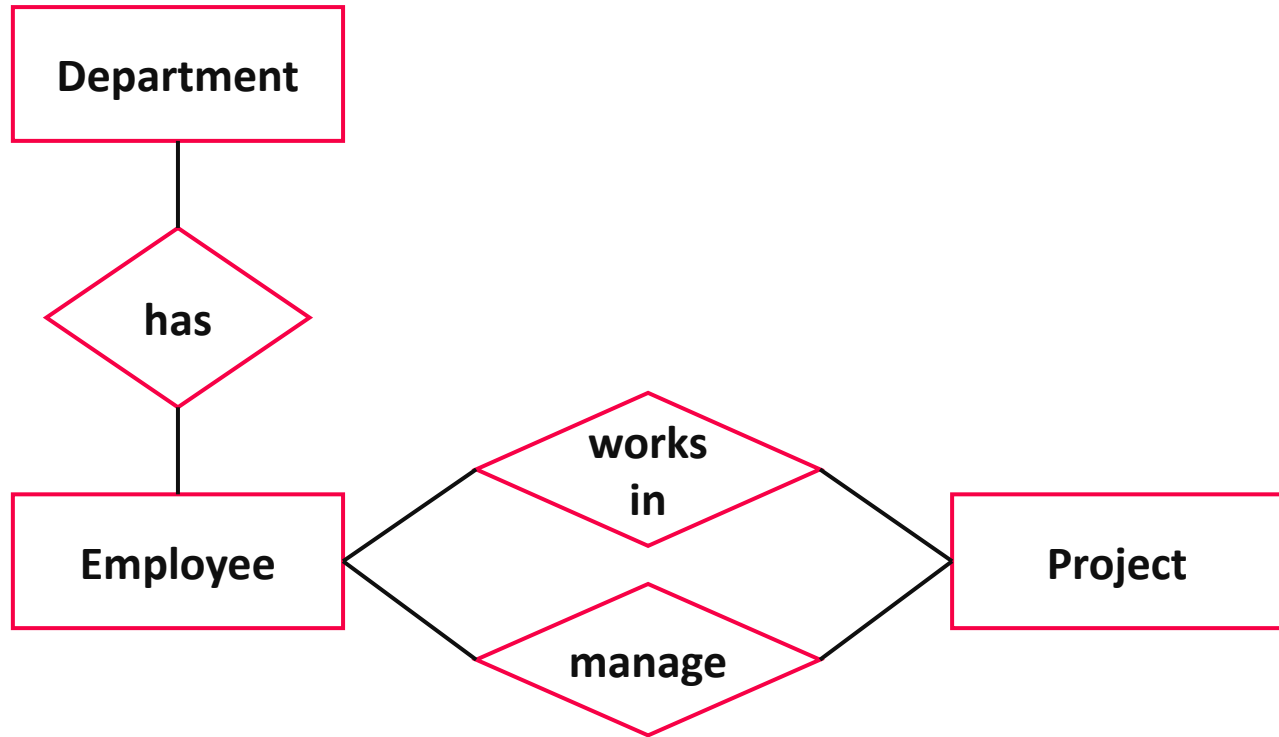
Multiplicity/Cardinality in Chen Notation



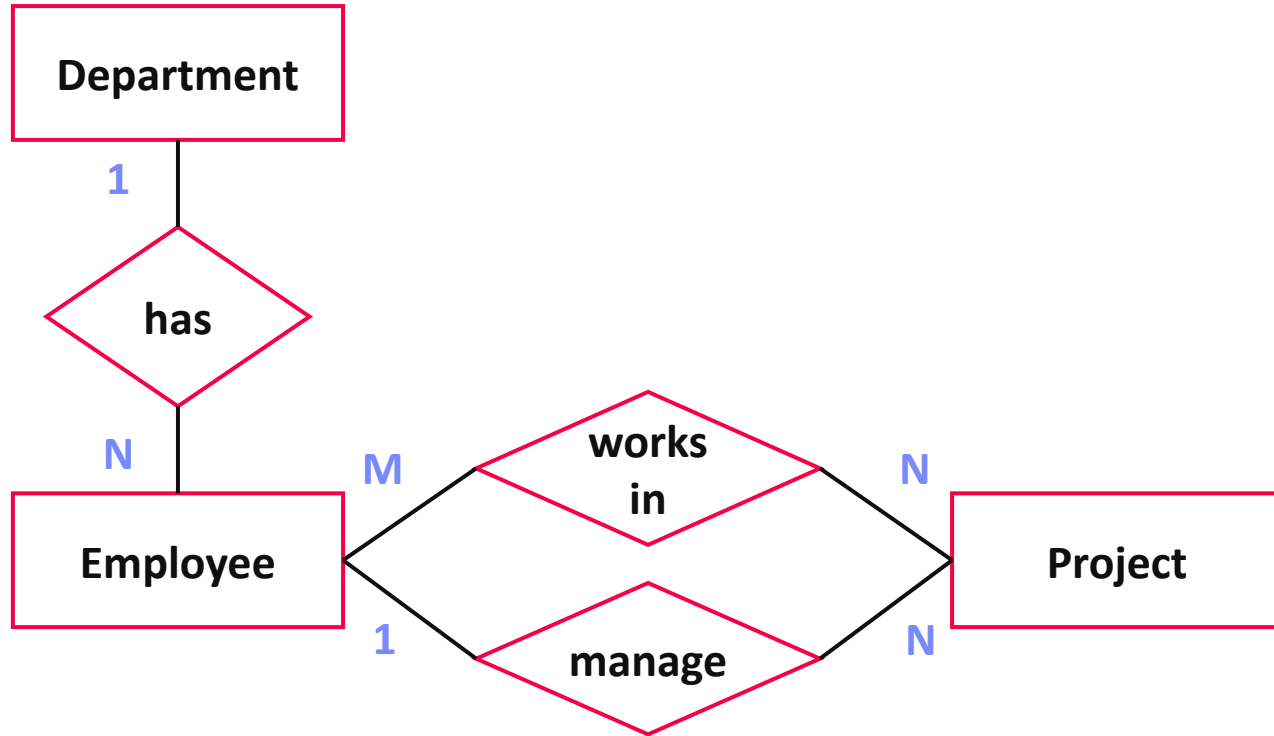
- N:M (many-to-many)
 - Each e1 relates to many e2 (0,1,...M)
 - Each e2 related to many e1 (0,1,...N)
 - e.g., 1 employee can work in M projects and 1 project can have N employees



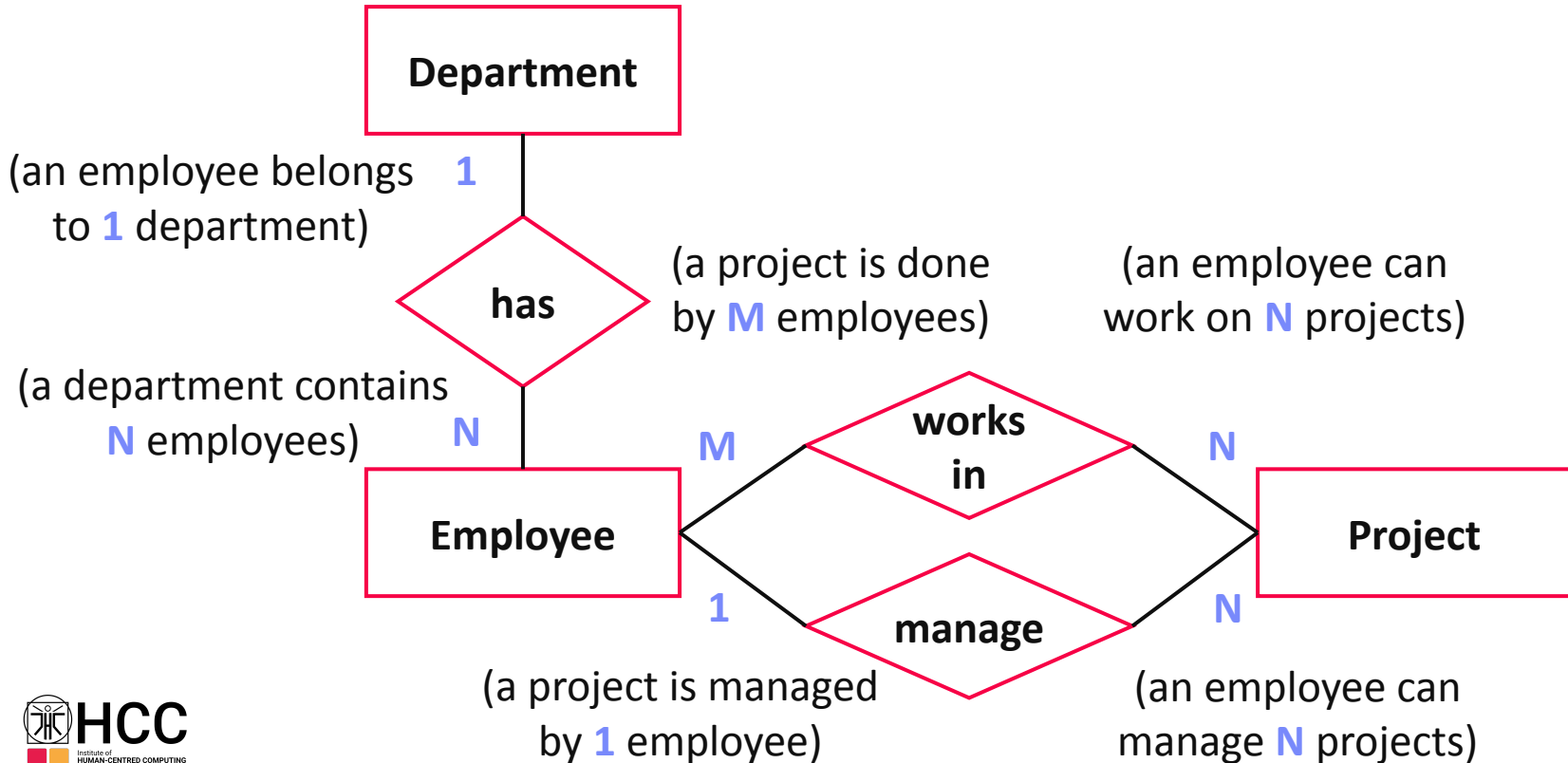
An EmployeeDB Example



An EmployeeDB Example



An EmployeeDB Example



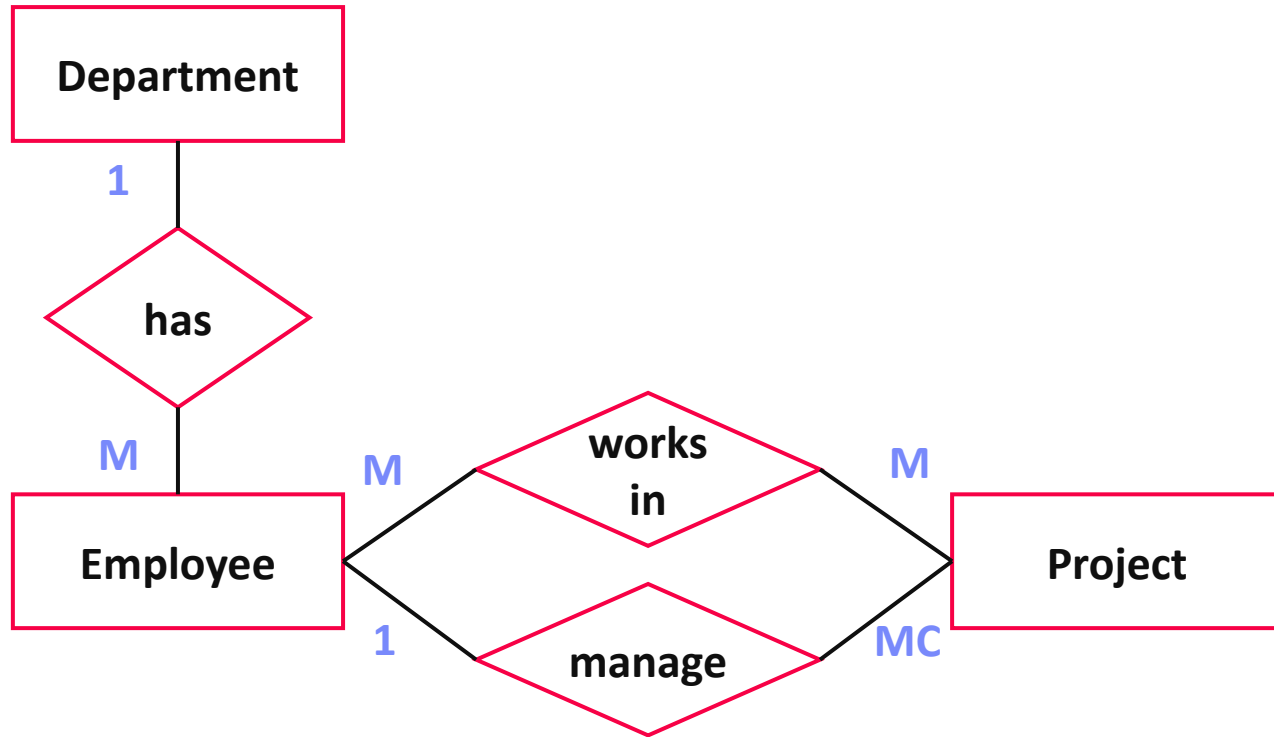
Multiplicity in **Modified Chen Notation**

- **Extension:** C (“choice”/“can”) to model 0 or 1, while 1 means exactly 1 and M means at least 1 (MC could also be 0)

Multiplicity in Modified Chen Notation

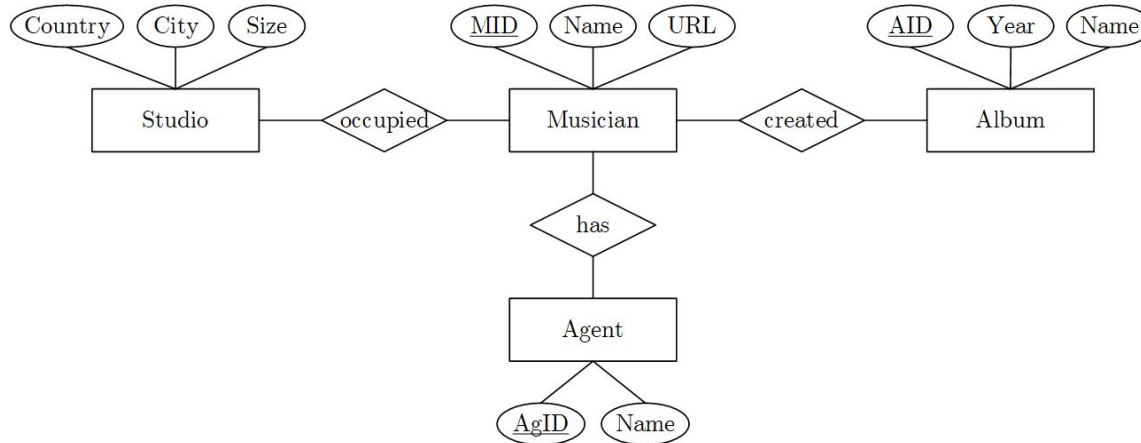
- Extension: C (“choice”/“can”) to model 0 or 1, while 1 means exactly 1 and M means at least 1 (MC could also be 0)
 - 1:1 – [1] to [1]
 - 1:C – [1] to [0 or 1]
 - 1:M – [1] to [at least 1]
 - 1:MC – [1] to [arbitrary many] (also 0 is possible!)
 - C:C – [0 or 1] to [0 or 1] **1:1 in Chen**
 - C:M – [0 or 1] to [at least 1]
 - C:MC – [0 or 1] to [arbitrary many] **1:N in Chen**
 - MC:C – [arbitrary many] to [0 or 1] **N:1 in Chen**
 - M:M – [at least 1] to [at least 1]
 - M:MC – [at least 1] to [arbitrary many]
 - MC:MC – [arbitrary many] to [arbitrary many] **M:N in Chen (Allows everything)**

An EmployeeDB Example in Modified Chen Not.

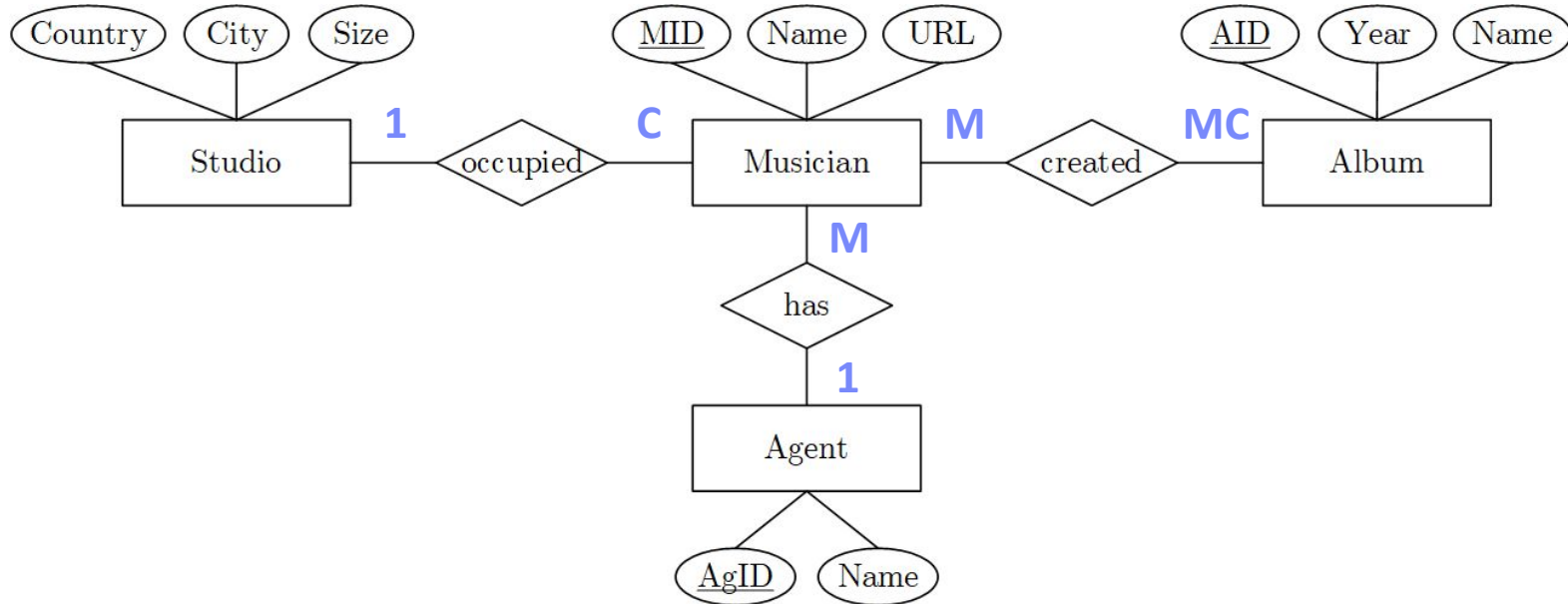


Another Example

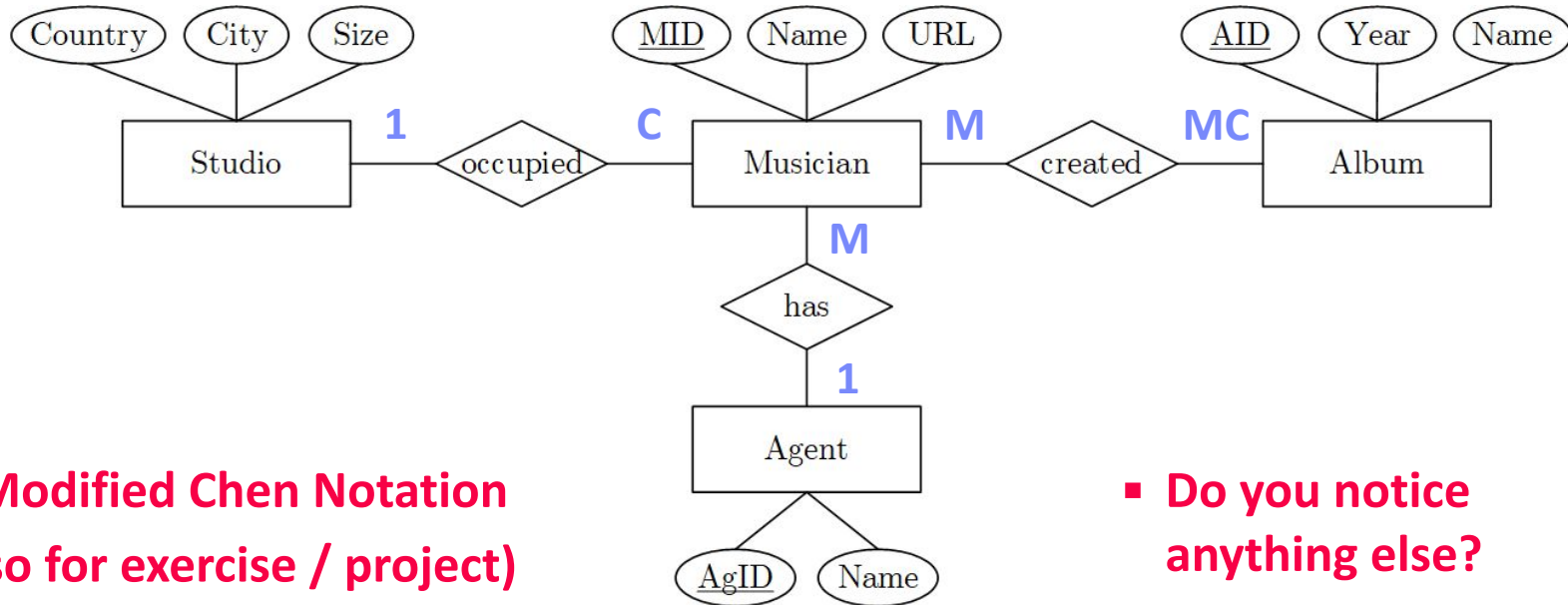
- Task: Cardinalities in **Modified-Chen Notation**
 - A musician might have created none or arbitrary many albums, and any album is created by at least one musician.
 - Every musician has exactly one agent, and an agent might be responsible for one to ten musicians.
 - Every musician occupies exactly one studio, and musicians never share a studio.



Another Example: Solution



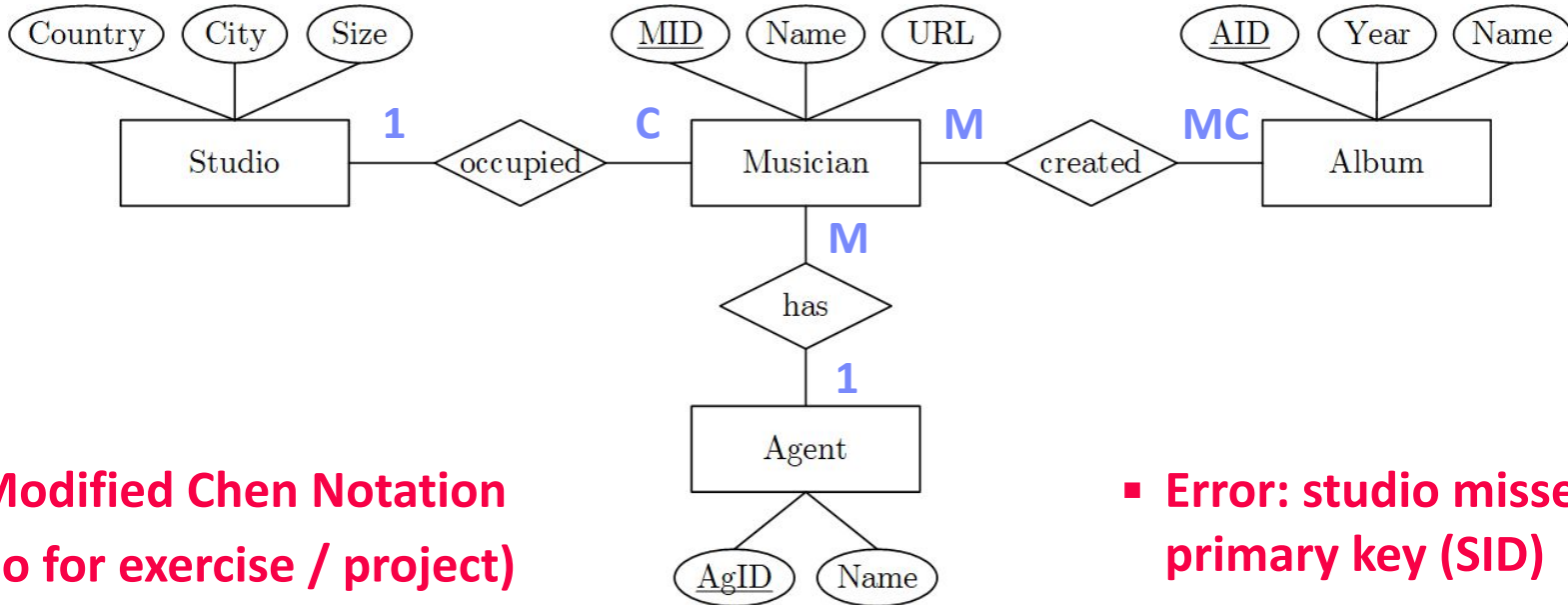
Another Example: Solution



▪ **Modified Chen Notation**
(also for exercise / project)

▪ **Do you notice anything else?**

Another Example: Solution



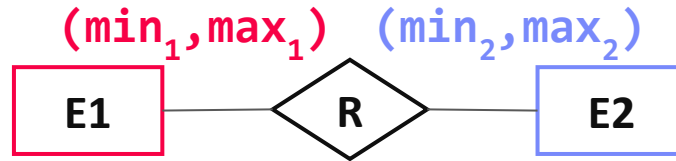
▪ **Modified Chen Notation**
(also for exercise / project)

▪ **Error: studio misses primary key (SID)**

(min,max)-Notation

Alternative Cardinality Notation

- **Indicate concrete min/max constraints**
(each entity is part of at least/at most x relationships)
- Modified chen --» **incoming** connection
- (min,max) --» **outgoing** connection
- **Wildcard** * indicates arbitrary many (i.e., MC)



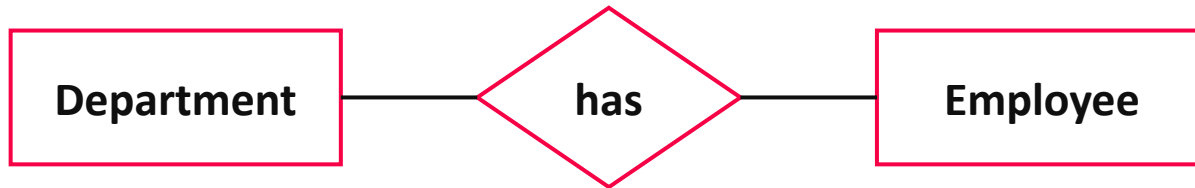
(min,max)-Notation

(each department has
1 – 70 employees)

(1,70)

(each employee in exactly
one department)

(1,1)



Modified Chen
notation

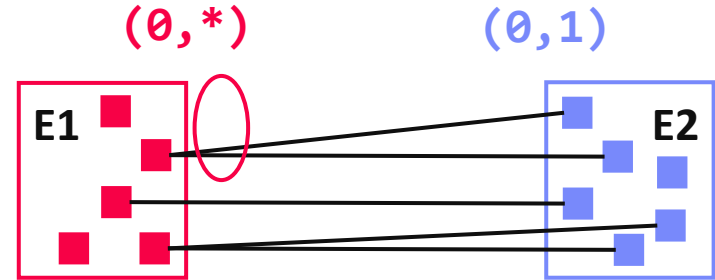
1

M

(for comparison)

(min,max)-Notation

- **Problem:** Where do these conflicting notations come from?
- Understanding (min, max)-Notation
 - **Focus on relationships!**
 - Describes **number of outgoing relationships** for each entity

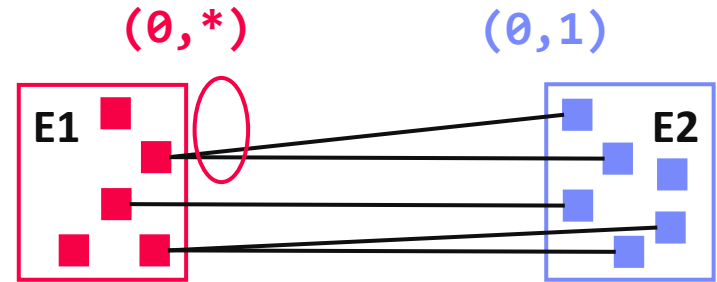


(min,max)-Notation

- **Problem:** Where do these conflicting notations come from?

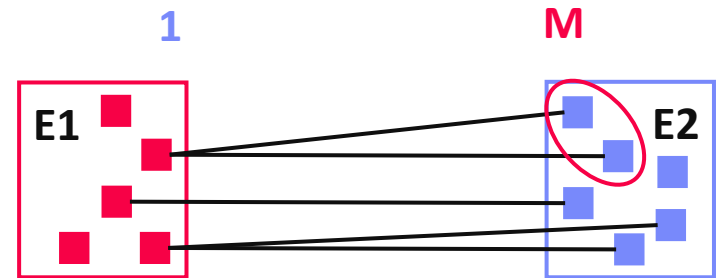
- Understanding (min, max)-Notation

- **Focus on relationships!**
- Describes number of outgoing relationships for each entity



- Understanding Chen- / Modified-Chen-Notation

- **Focus on entities!**
- Describes number of target entities (over relationships) for each entity

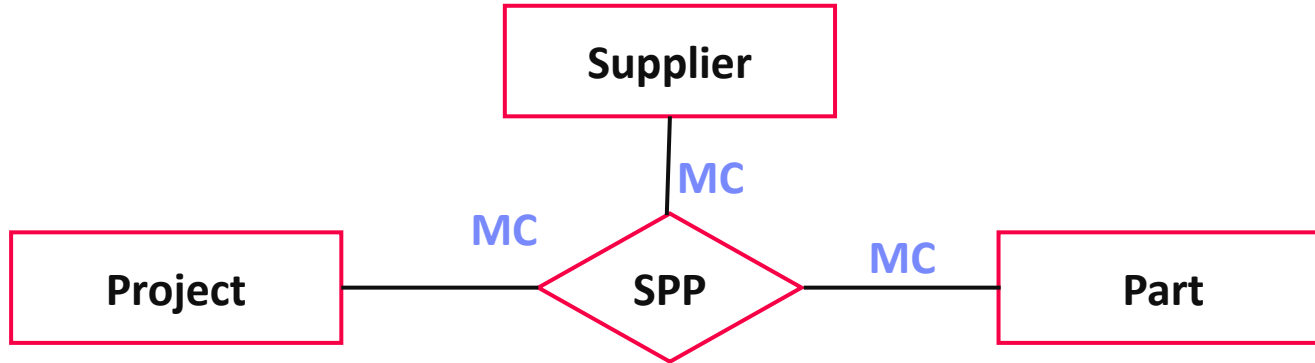


N-ary Relationships

- Relationship among multiple entities
- In some cases N-ary relationship can be **converted to binary relationships**

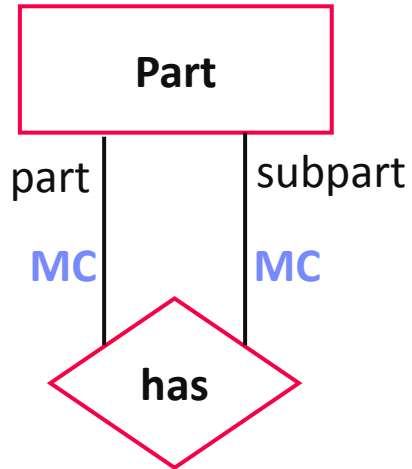
N-ary Relationships

- Relationship among multiple entities
- In some cases N-ary relationship can be converted to binary relationships



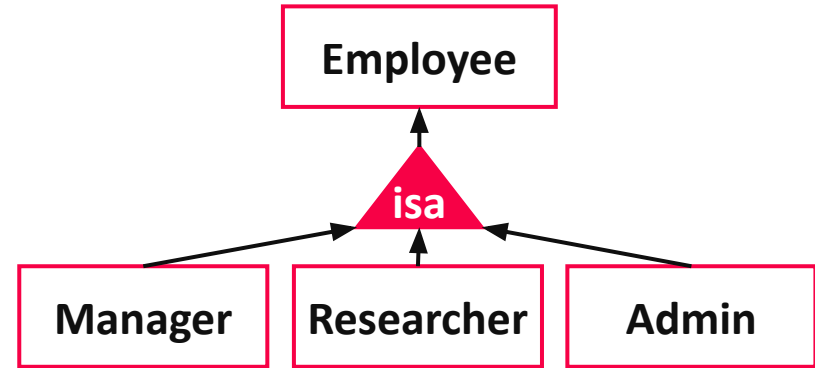
Recursive Relationships

- Definition
 - Recursive relationships are **relations between entities of the same type**
 - Use roles to differentiate cardinalities



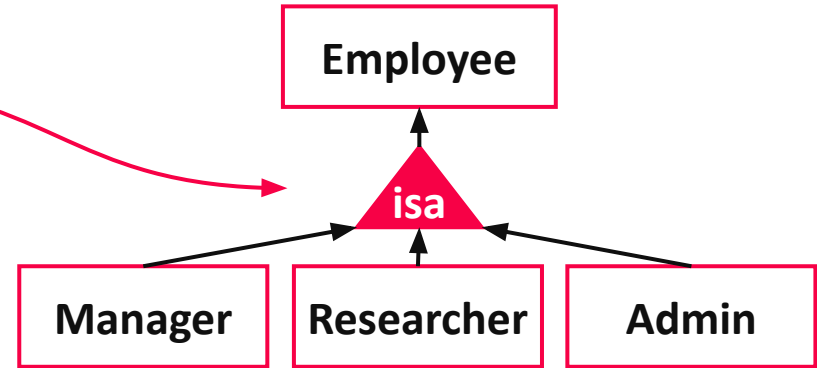
Specialization via Subclasses

- **Tree of specialized entity types**
Graphical symbol: triangle
- Each **entity of subclass** is **entity of superclass**, but **not vice versa**



Specialization via Subclasses

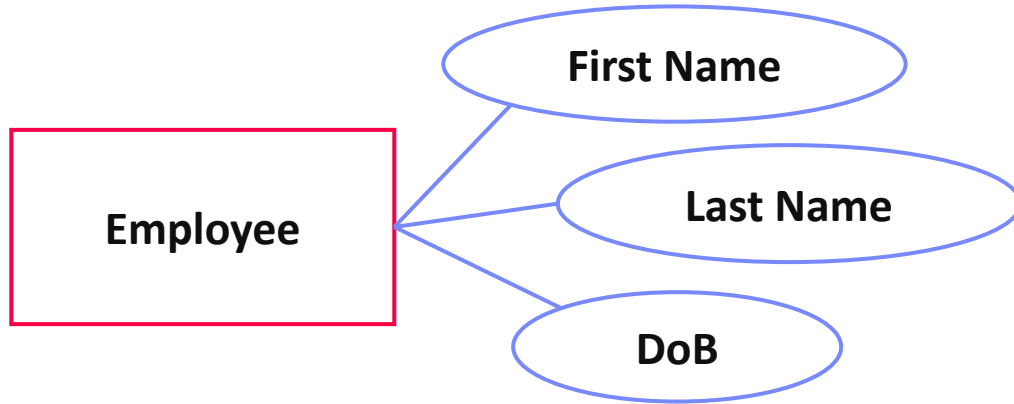
- **Tree of specialized entity types**
Graphical symbol: **triangle**
- Each **entity of subclass** is **entity of superclass**, but **not vice versa**
- No multi-inheritance



Types of Attributes

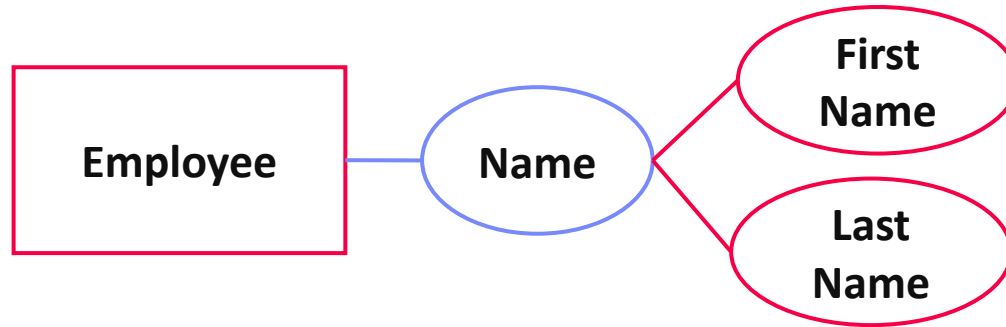
Atomic Attributes

- Basic, **single-valued** attributes



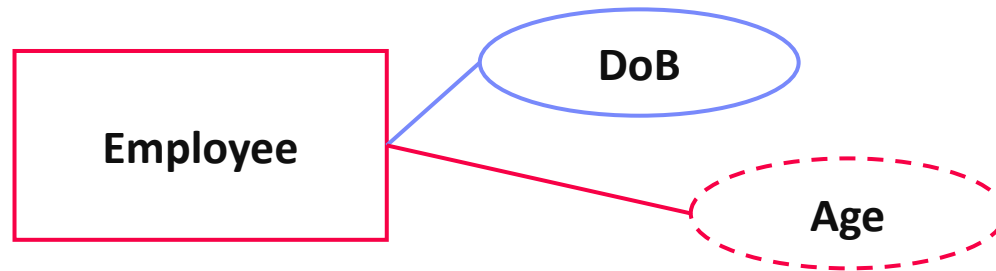
Composite Attributes

- Attributes as **structured data types**
- Can be represented as a **hierarchy**



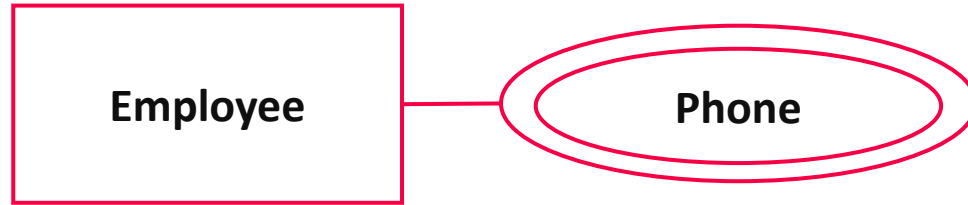
Derived Attributes

- Attributes **derived from other data**
- Examples: Number of employees in dep, employee age, ...



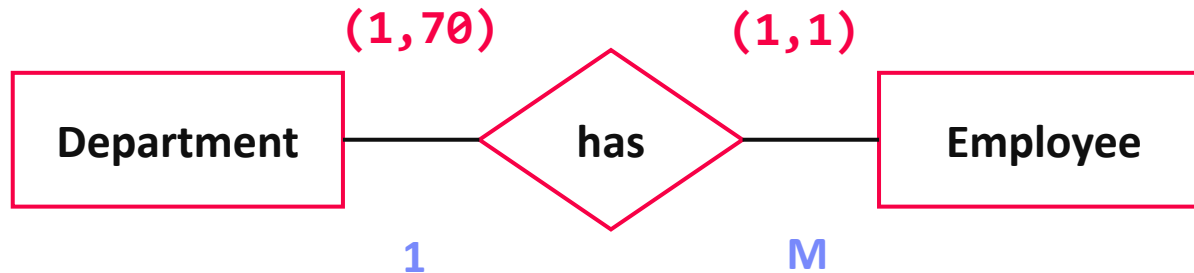
Multi-valued Attributes

- Attributes with list of similar entries



Design Decisions - Notations

- Which notations to use (Chen, Modified Chen, (min,max)-notation)?
 - **Modified Chen for exercise and project!**



Design Decisions - Entities

- What are the **entity types**?
- What are the **attributes of each entity type**?
- What are **key attributes** (one or many)?
- What are **weak entities** (with partial keys)?



Employee

Design Decisions - Relationships

- What are the **relationship types between entities** (binary, n-ary)?
- What are the **attributes of each relationship type**?
- What are the **cardinalities**?

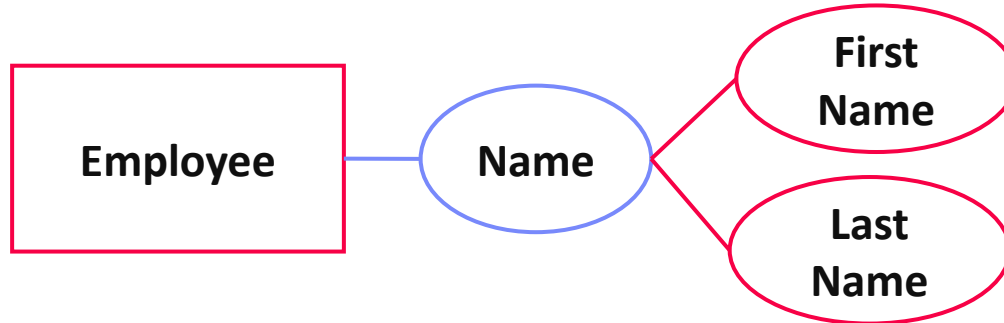
Design Decisions - Relationships

- What are the **relationship types** between entities (binary, n-ary)?
- What are the **attributes of each relationship type**?
- What are the **cardinalities**?

- **Stick to binary** relationships whenever possible
 - Also for exercise and project
 - BUT **sometimes** n-ary relationship make sense

Design Decisions - Attributes

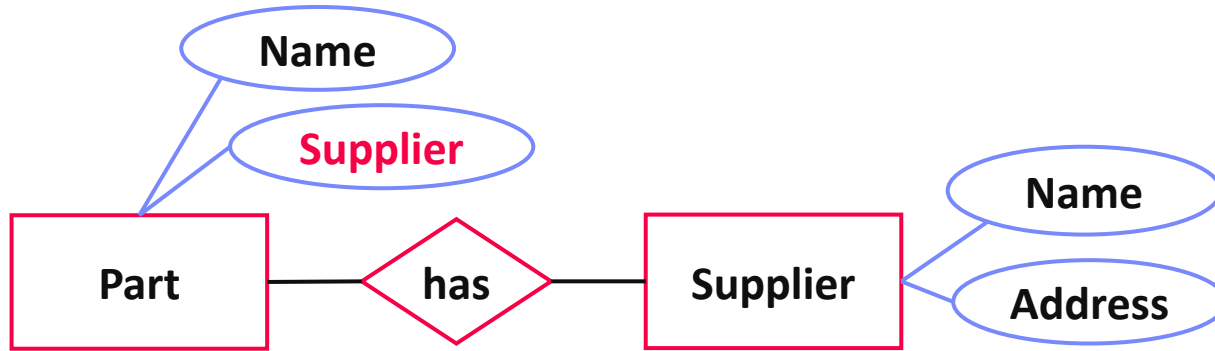
- What are atomic, composite, multi-valued, or derived attributes?
 - Use **atomic** whenever possible



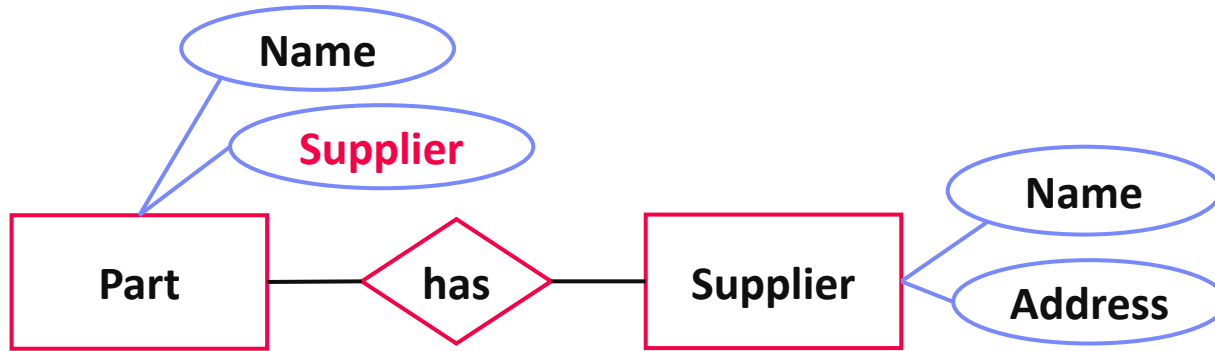
Design Decisions

Examples of Poor Choices

Example 1

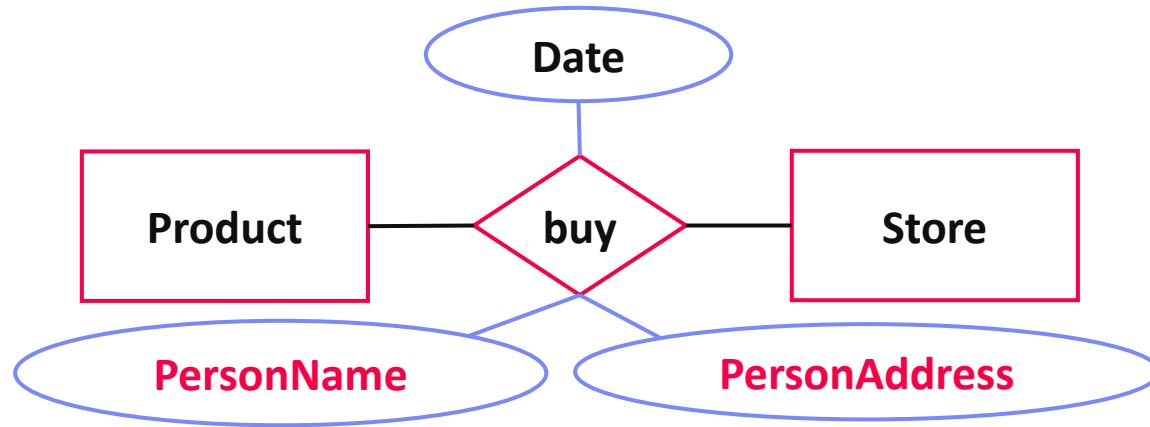


Redundant attributes

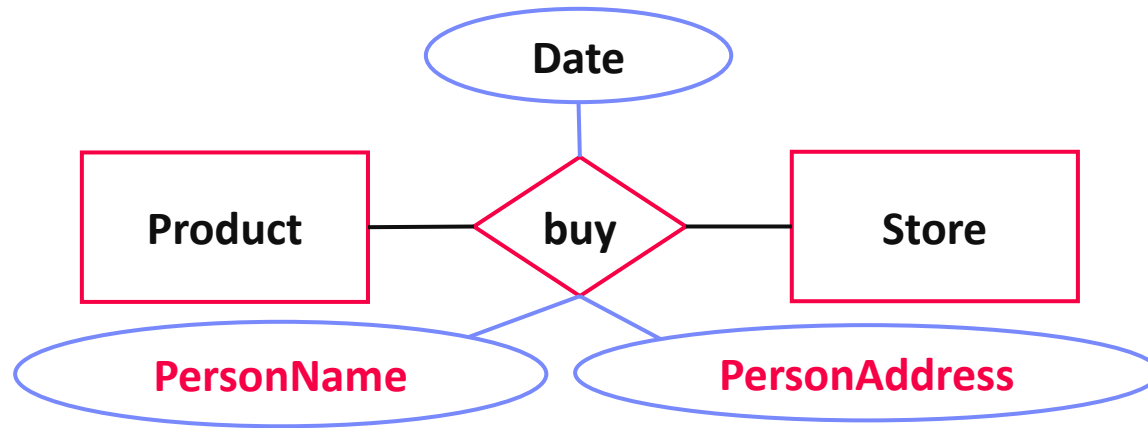


- **Redundant supplier name** in Part and Supplier

Example 2

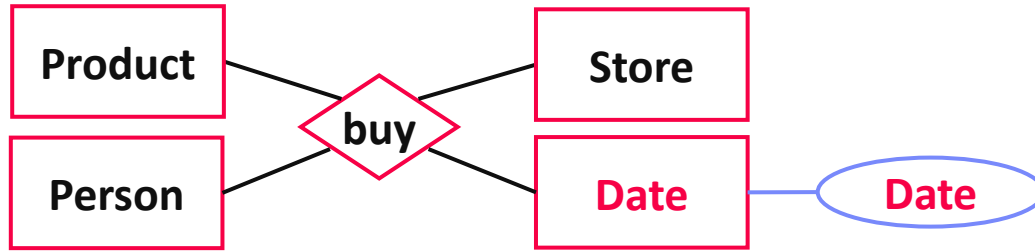


Redundant information

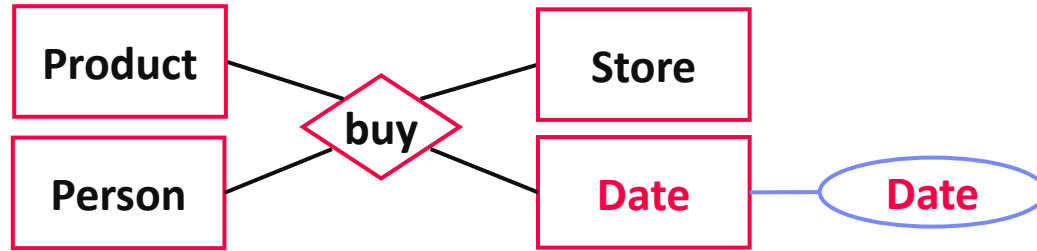


- **Missing person entity type**
--» redundancy per purchase

Example 3

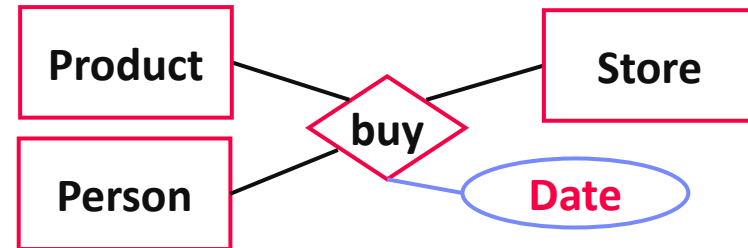


Unnecessary Complexity



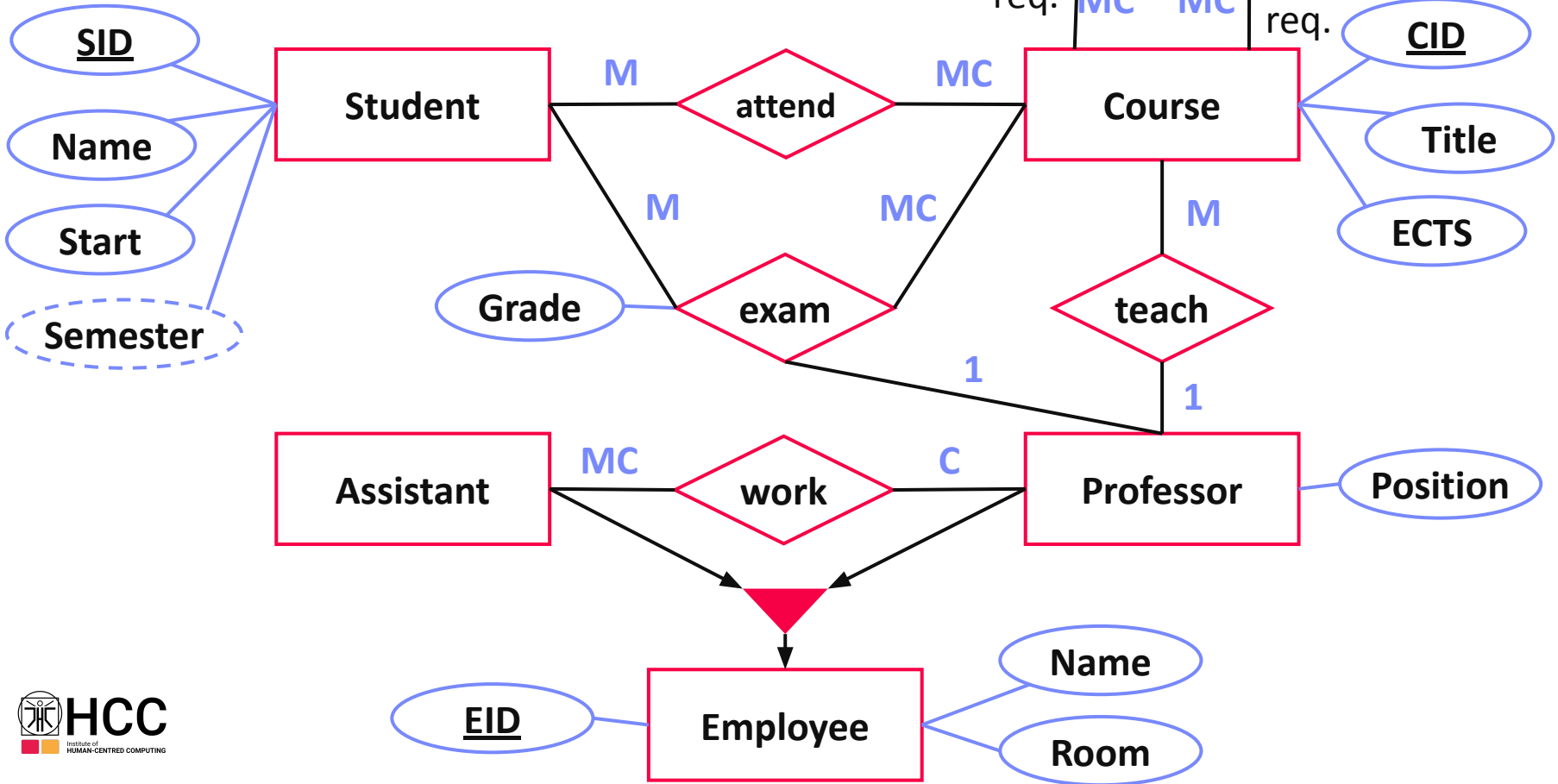
- **Unnecessary entity type Date**

--» Better to store it with the relationship „buy“



A UniversityDB Example

- Discourse of Real Mini World
 - **Students** (with SID, name, start date, and semester) attend **courses** (CID, title, ECTS), and take graded exams per course
 - A **course** may have other courses as prerequisites
 - **Professors** teach and grade courses and have positions (e.g. Dean of Studies), **assistants** may work for professors
 - Both professors and assistants are university **employees** (EID, name, and room number); **professors** also have a position (e.g. Dean of Studies)
- Task: **Create an ER diagram** in Modified Chen notation
 - Include entity types, relationship types, attributes, and generalizations
 - Mark primary keys, roles for recursive relationships, and derived attributes



Exercise 1 – Data Modeling

Deadline Tue 18th of November 2025 (23:59)

Exercise 1 (25 points)

- **Gift cards review dataset**

- This dataset contains Amazon gift card reviews collected in 2023.
- Datasets are shared for research and teaching purposes
- Original dataset: <https://www.kaggle.com/datasets/qwaazs/amazon-gift-and-card-reviews>

- **Task 1: ER Modeling (15 points)**

- **Task 2: Mapping ER Model to Relational Model (10 points)**



Exercise 1

Task 1.1 ER Modeling (15/25 points)

Create an **ER diagram in Modified Chen (MC) notation** – including entity types, relationship types, attributes, cardinalities, and keys (create surrogate keys if natural unique identifiers are missing) – for managing data stored on a platform selling gift cards of various shops.

Please, use the digital tool **draw.io** for designing the ER diagram. We recommend draw.io since it has a toolbar for ER elements (among others).

Exercise 1

- **Details**

- The database contains data on **users** registered at the platform. Each **user** is described by a unique ID, gender, and age.
- There are also **gift cards** listed on the platform. A **gift card** on the platform has an ID, title, description, and price predefined for it at the moment of listing. In addition, the database stores the number of ratings submitted by **users** and the average rating of a **gift card**.
- **Gift cards** are assigned to the **stores** they were issued by. The dataset contains data on the ID and name of each **store**. Every **gift card** is issued exclusively by a particular **store**. At the same time, not every **store** offers a **gift card**. But none of them are limited in the number of different types of **gift cards** they issue.

Exercise 1

- Details (cont)
 - In order to provide filtering functionality during the search to its **users** the platform assigns 1-3 **categories** to every **gift card**. The platform ensures that there is at least one **gift card** in each **category** listed at all times. A **category** is described by an ID and a name.
 - Some **users** post **reviews** about the **gift cards** they buy on the platform. There is no limit on the number of **reviews** a **user** can post on the platform. Each **review** includes an ID, a title, the actual text of the **review** and a **user** rating (score on a scale 1.0-5.0). The platform also automatically assigns a timestamp to a **review**.
 - A **gift card** can be subject to many **reviews** and at this point all the items listed on the platform were reviewed at least once. It should also be noted that in each **review** only a single **gift card** can be discussed by the **user**.

Exercise 1

- All details in Exercise Description
- Use only Modified Chen Notation: **M, MC, C, 1.**
- Deliverable 1: **ERDiagram.pdf (1 page PDF)**
- **CREATE TABLE statements (Task 2, see next week)**

Example

Disclaimer: This section contains an example. The exercise to be solved to approve the course can be found in the Teach Center.

Task 1.1

Task 1.2

PostgreSQL & pgAdmin

Example T1.1 (similar to our exercise)

Create an ER diagram in Modified Chen notation for managing data stored on the Formula 1 Championship Management System

The database should capture the essential information of the following statements:

- The database contains data on **drivers** participating in the F1 Championship. Each driver has a unique ID (e.g., 44 for Lewis Hamilton), name, nationality, and date of birth.
- **Teams** participate in the championship, and each team has exactly two drivers. A team is described by its name, country, and foundation year.
- Each **Grand Prix** is an event held in a specific country and has attributes such as circuit name, date, and number of laps. Each Grand Prix can have multiple drivers competing, and each driver may participate in multiple Grand Prix events.



Example T1.1 (similar to our exercise)

Create an ER diagram in Modified Chen notation for managing data stored on the Formula 1 Championship Management System

Step 1: Identify entities and attributes

Entity 1: Driver

- ID (Primary Key)
- Name
- Nationality
- Date of Birth

Entity 2: Teams

- Team Name (Primary Key)
- Country
- Foundation Year

Entity 3: Grand Prix

- Grand Prix ID (Primary Key)
- Circuit Name
- Country
- Date
- Number of Laps

Example T1.1 (similar to our exercise)

Create an ER diagram in Modified Chen notation for managing data stored on the Formula 1 Championship Management System

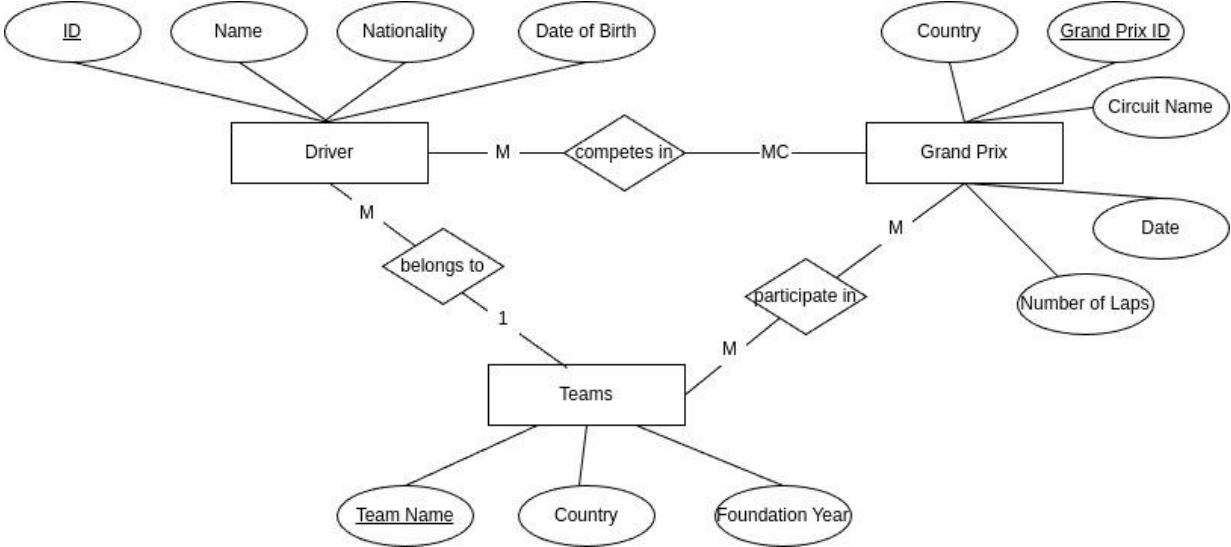
Step 2: Relationships and cardinalities

| Relationship | Cardinality | Explanation |
|-------------------------------------|-------------|---|
| Driver - Belongs To - Team | (M:1) | Each driver belongs to one team, and each team has exactly two drivers. |
| Driver - Competes In - Grand Prix | (M:MC) | A Grand Prix has multiple drivers competing, and a driver may participate in multiple Grand Prix races (or none). |
| Team - Participates In - Grand Prix | (M:M) | Each team participates in multiple Grand Prix races, and each Grand Prix has multiple teams competing. |

Example T1.1 (similar to our exercise)

Create an ER diagram in Modified Chen notation for managing data stored on the Formula 1 Championship Management System

Step 3: ER Map



Example T1.2 (similar to our exercise)

Create a relational schema for the ER diagram designed in Task 1.1 and bring it into third normal form. This schema should include the relations and typed attributes, as well as all primary and foreign keys. Then, use the PostgreSQL Data Base Management System to create a SQL script (CreateSchema.sql) with CREATE TABLE statements.

Step 1:

- Download installer via: <https://www.postgresql.org/download/>
- Setup PostgreSQL in your Laptop (See Tutorial in Teach Center)

Example T1.2 (similar to our exercise)

Step 2: create your schema and normalize (1NF, 2NF, **3NF**)

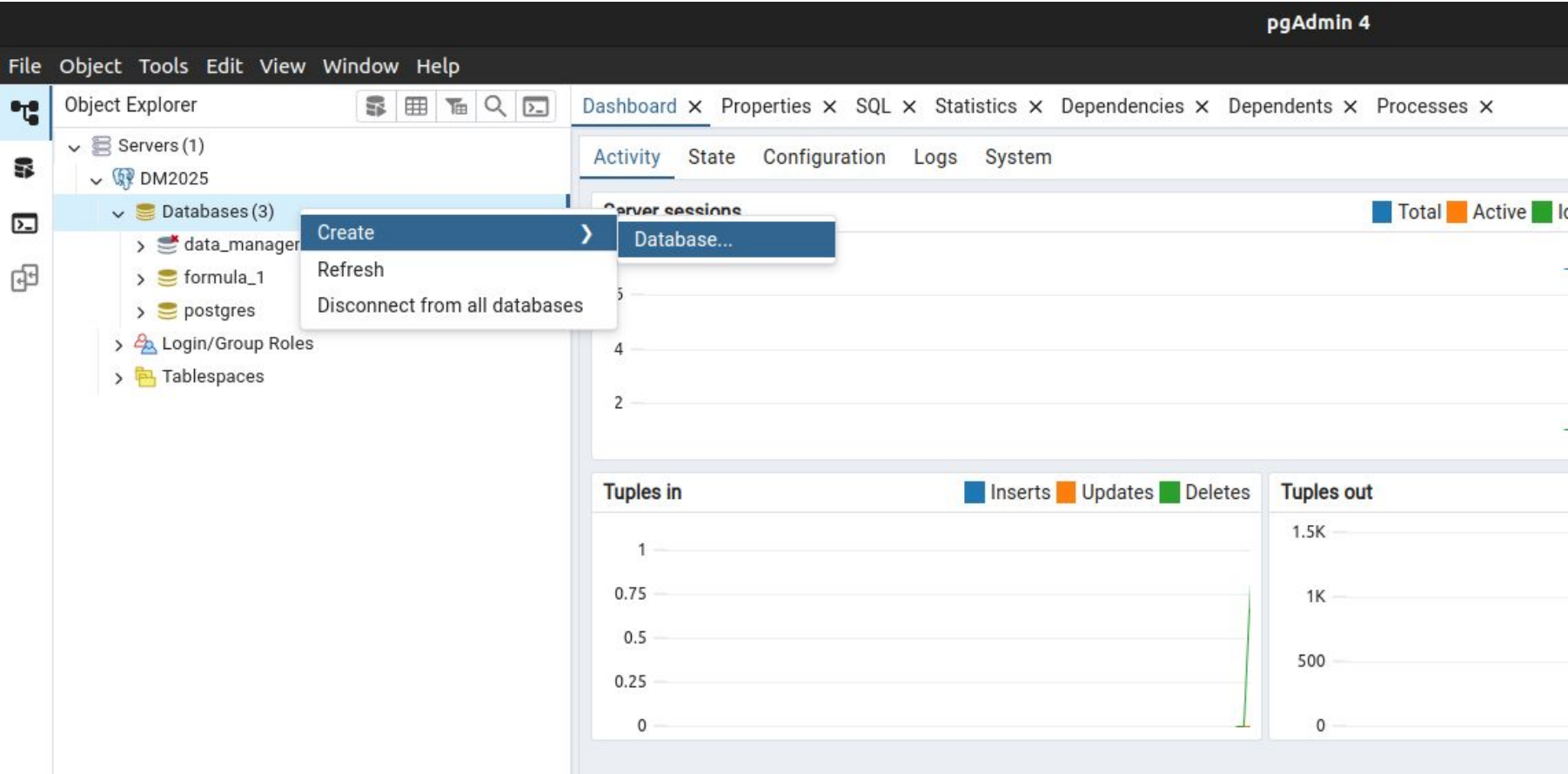
(CREATE Table Statements)

Example T1.2 (similar to our exercise)

Step 3: Implement your schema in PostgreSQL



Example T1.2 (similar to our exercise): Create DB



The screenshot shows the pgAdmin 4 interface. In the Object Explorer on the left, the 'Databases (3)' folder is expanded, and the 'Create' context menu is open over it. The 'Create Database...' option is highlighted. The main window displays the 'Server sessions' table and two performance graphs: 'Tuples in' and 'Tuples out'.

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x

Servers (1)
 DM2025
 Databases (3)
 data_manager
 formula_1
 postgres
 Login/Group Roles
 Tablespaces

Activity State Configuration Logs System

Server sessions

Total Active

Tuples in Inserts Updates Deletes

Tuples out

Example T1.2 (similar to our exercise): Create DB

Object Explorer

- Servers (1)
 - DM2025
 - Databases (3)
 - data_management
 - formula_1
 - postgres
 - Login/Group Roles
 - Tablespaces

Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x

Activity State Configuration Logs System

Server sessions Total Active Idle Transactions per second

6
4
2

Tuples in

1
0.75
0.5
0.25
0

Returned Block I/O

150
100
50
0

Create - Database

General Definition Security Parameters Advanced SQL

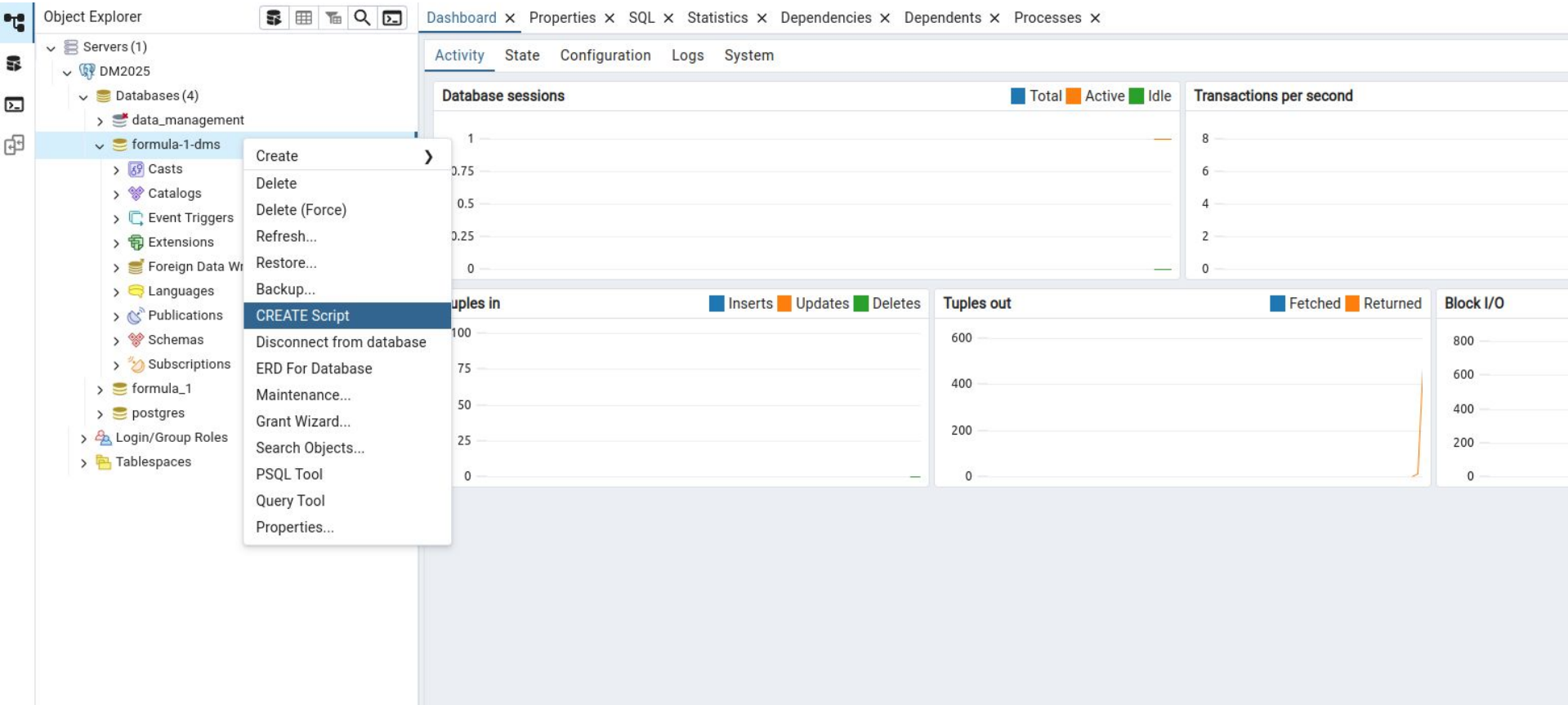
Database

Owner

Comment

Close Reset Save

Example T1.2 (similar to our exercise): Create Schema



The screenshot displays the PostgreSQL Enterprise Manager interface. On the left, the Object Explorer shows a tree view of servers and databases. The 'formula-1-dms' database is selected, and a context menu is open over it, with 'CREATE Script' highlighted. The main panel shows performance metrics for the selected database.

Database sessions

| Session ID | Total | Active | Idle |
|------------|-------|--------|------|
| 1 | 1 | 0 | 1 |

Transactions per second

| Transaction ID | Value |
|----------------|-------|
| 1 | 8 |

Tuples in

| Tuple ID | Inserts | Updates | Deletes |
|----------|---------|---------|---------|
| 1 | 0 | 0 | 0 |

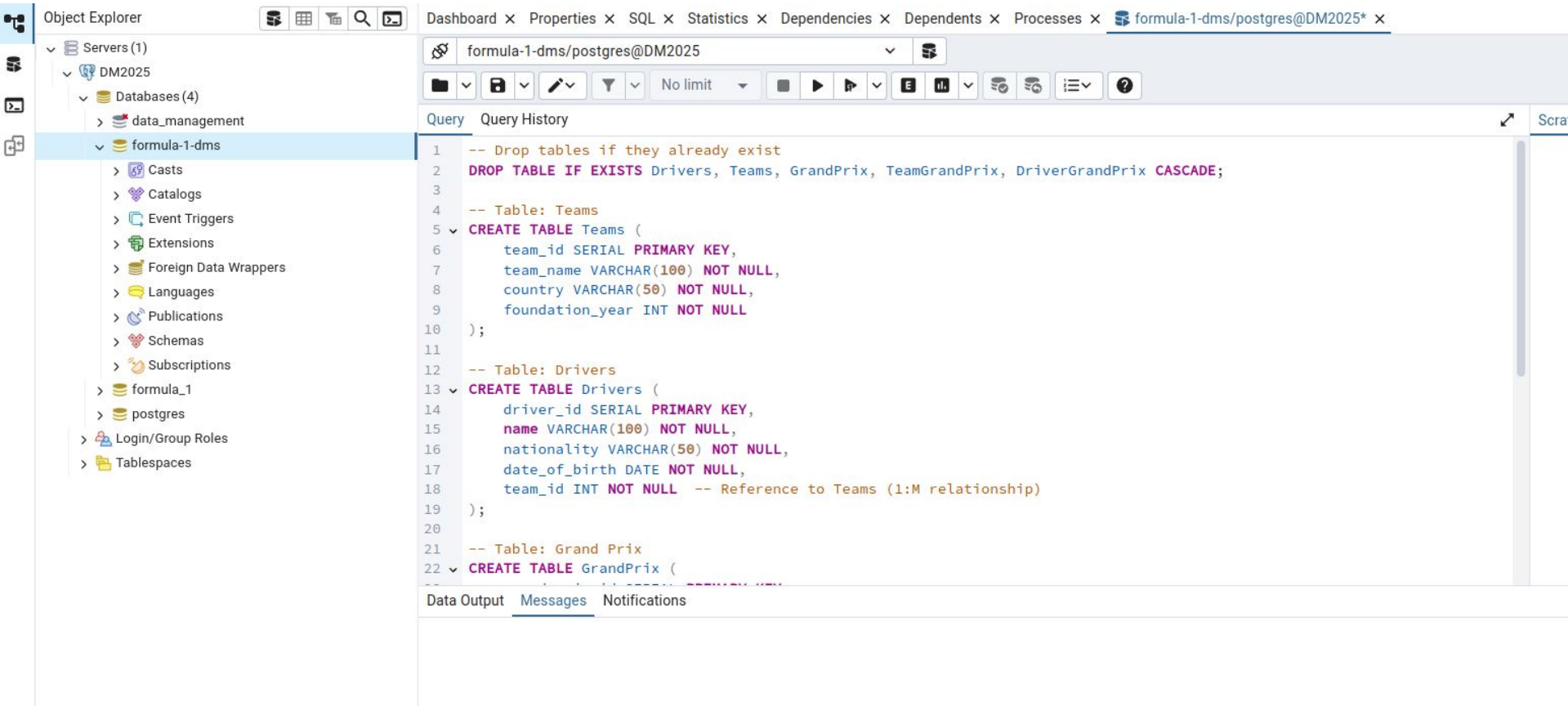
Tuples out

| Tuple ID | Fetches | Returns |
|----------|---------|---------|
| 1 | 0 | 0 |

Block I/O

| Block I/O ID | Value |
|--------------|-------|
| 1 | 0 |

Example T1.2 (similar to our exercise): Create Schema



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server hierarchy: Servers (1) > DM2025 > Databases (4) > data_management > formula-1-dms. The main pane displays a SQL query for creating tables in the formula-1-dms database.

Query

```

1  -- Drop tables if they already exist
2  DROP TABLE IF EXISTS Drivers, Teams, GrandPrix, TeamGrandPrix, DriverGrandPrix CASCADE;
3
4  -- Table: Teams
5  CREATE TABLE Teams (
6      team_id SERIAL PRIMARY KEY,
7      team_name VARCHAR(100) NOT NULL,
8      country VARCHAR(50) NOT NULL,
9      foundation_year INT NOT NULL
10 );
11
12 -- Table: Drivers
13 CREATE TABLE Drivers (
14     driver_id SERIAL PRIMARY KEY,
15     name VARCHAR(100) NOT NULL,
16     nationality VARCHAR(50) NOT NULL,
17     date_of_birth DATE NOT NULL,
18     team_id INT NOT NULL -- Reference to Teams (1:M relationship)
19 );
20
21 -- Table: Grand Prix
22 CREATE TABLE GrandPrix (

```

Data Output Messages Notifications

Conclusions

- **Summary**
 - **Database design**
 - **Entity-relationship (ER) models and diagrams**
 - **Exercise 1 - T1.1**

Conclusions

- **Summary**
 - Database design
 - Entity-relationship (ER) models and diagrams
 - Exercise 1 - T1.1.

- **Next week**
 - Relational databases and normalization
 - Bit SQL for Exercise 1
 - Exercise 1 - T1.2)