Database Systems
Part-2

Relation, FD, and Keys

Arijit Khan

Associate Professor
Department of Computer Science
Aalborg University, Denmark

arijitk@cs.aau.dk

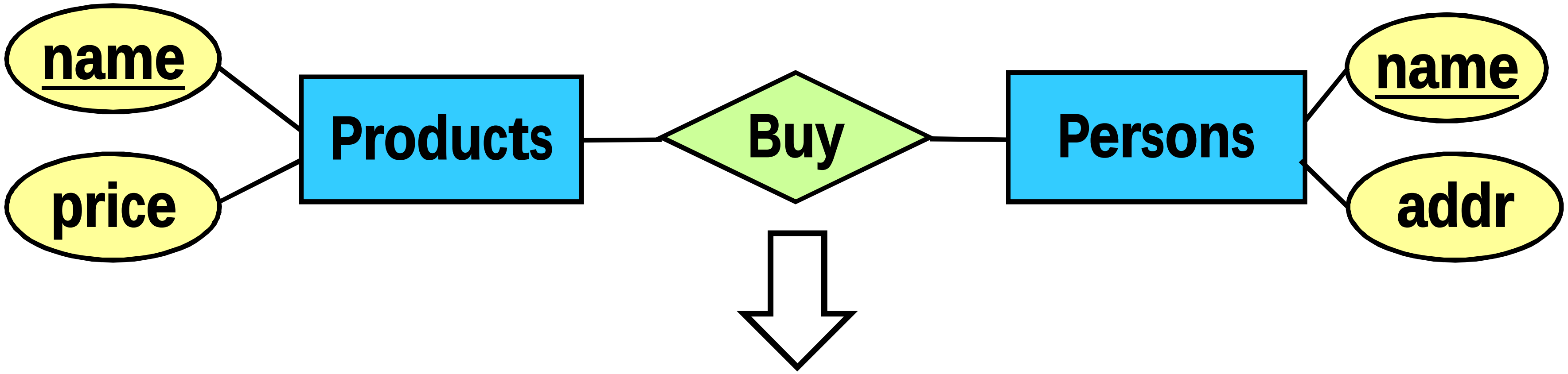
Schedule for first half

Intro & ER	Sep 5 Monday (12:30-14:15), Lecture - FRB 7H	No exercise
Relation, FD, Keys	Sep 12 Monday (12:30-14:15), Lecture - FRB 7H	Exercise-1 on Intro & ER (14:30-16:30)
Normalization	Sep 19 Monday (12:30-14:15), Lecture - FRB 7H	Exercise-2 on Relation, FD, Keys (14:30-16:30)
Relational Algebra	Exercise-3 on Normalization (8:15-10:00)	Sep 23 Friday (10:15-12:00), Lecture - NOVI8
Release of Self-Study-1	Sep 26 Monday, No lecture, No exercise	
SQL – Part I	Oct 3 Monday (12:30-14:15), Lecture - FRB 7H	Exercise-4 on Relational Algebra (14:30-16:30)
Feedback of Self-Study-1	Oct 10 Monday (12:30-14:15), During lecture - FRB 7H	Exercise-5 on SQL – Part I (14:30-16:30)

Ask Questions!

> The important thing is not to stop questioning.
>
> Albert Einstein

ER Diagram → Relational Schema

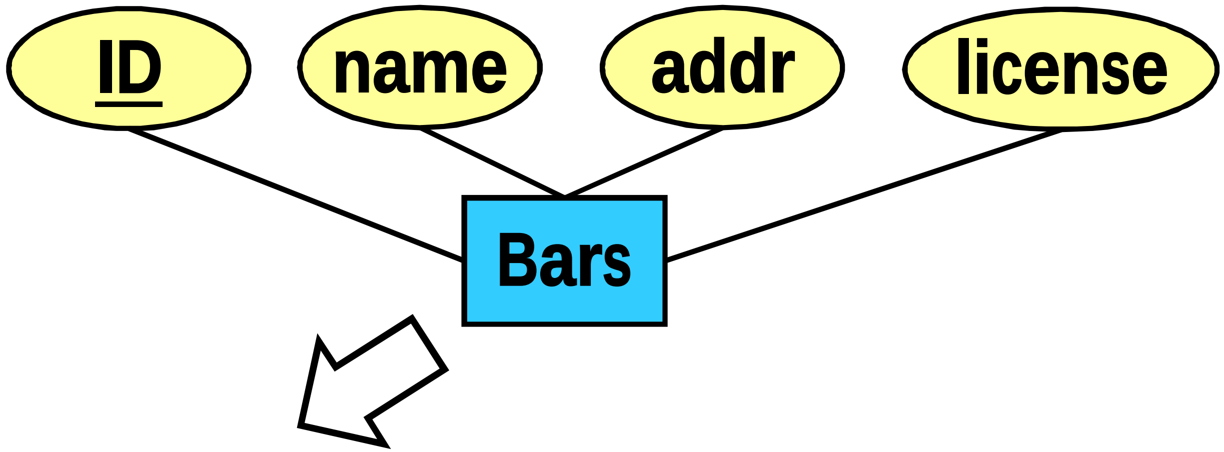


- Products (<u>name</u>, price)
- Persons (<u>name</u>, addr)
- Buy (<u>product_name</u>, <u>person_name</u>)
- Terminology
  - A relation schema = the name of a table + names of its attributes
  - A database schema = a set of relation schemas

ER Diagram → Relational Schema

- General rules:
  - Each entity set becomes a relation
  - Each many-to-many relationship becomes a relation
- Special treatment needed for:
  - Weak entity sets
  - Subclasses
  - Many-to-one and one-to-one relationships

Entity Set → Relation

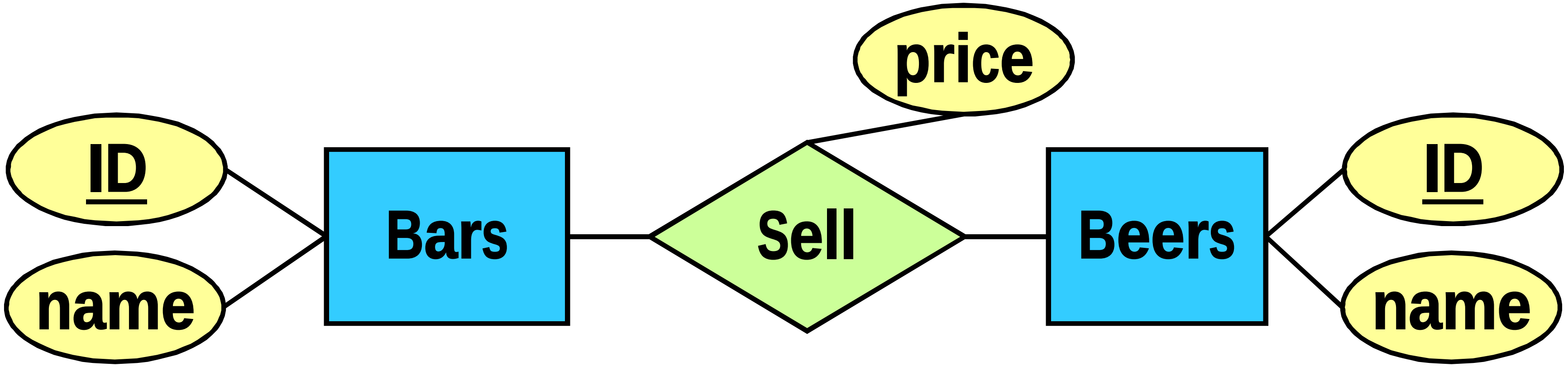


Bars

<u>ID</u>	name	addr	license
...	...	...	...

- Each entity set is converted into a relation that contains all its attributes
- The key of the relation = the key of the entity set

Many-to-Many Relationship → Relation

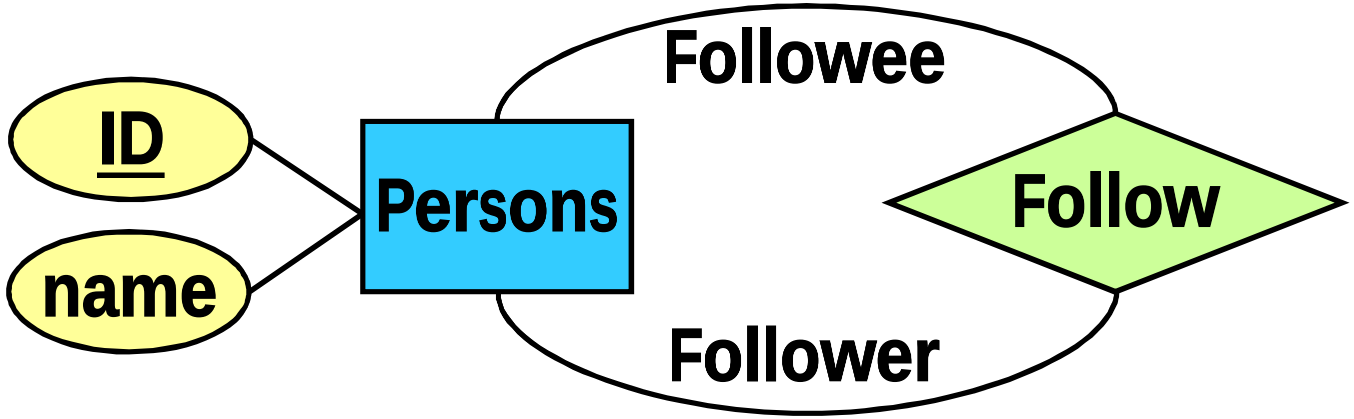


- Converted into a relation that contains
  - all keys of the participating entity sets, and
  - the attributes of the relationship (if any)
- Key of relation = Keys of the participating entity sets

Sell

<u>Bars-ID</u>	<u>Beers-ID</u>	price
…	…	…

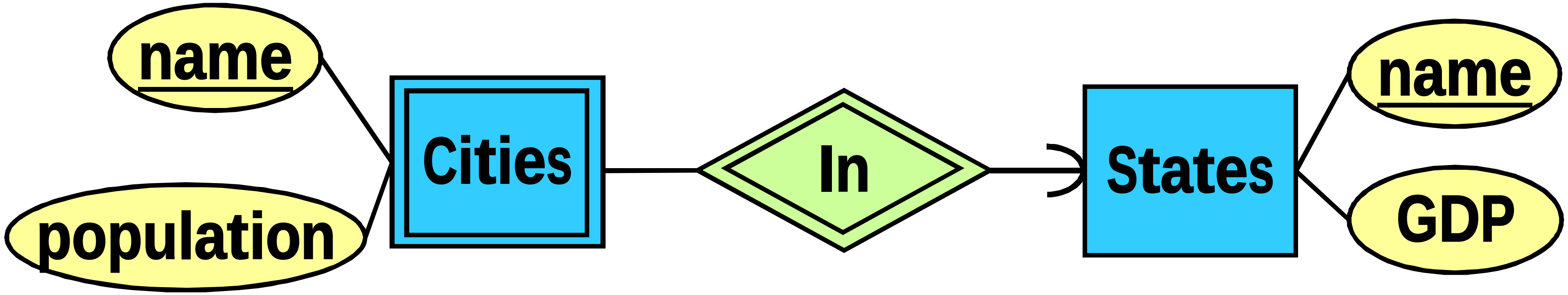
Many-to-Many Relationship → Relation



- If an entity is involved multiple times in a relationship
  - Its key will appear in the corresponding relation multiple times
  - The key is re-named according to the corresponding role

Follow	Follower-ID	Followee-ID
	...	...

Weak Entity Set → Relation

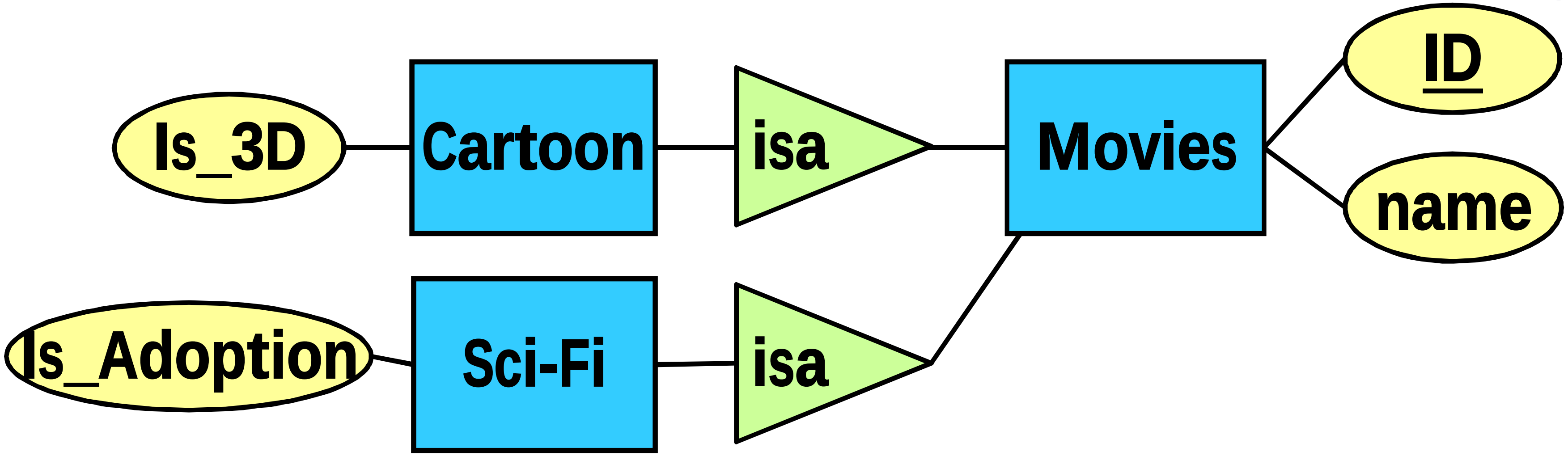


- Each weak entity set is converted to a relation that contains
  - all of its attributes, and
  - the key attributes of the supporting entity set
- The supporting relationship is ignored

Cities

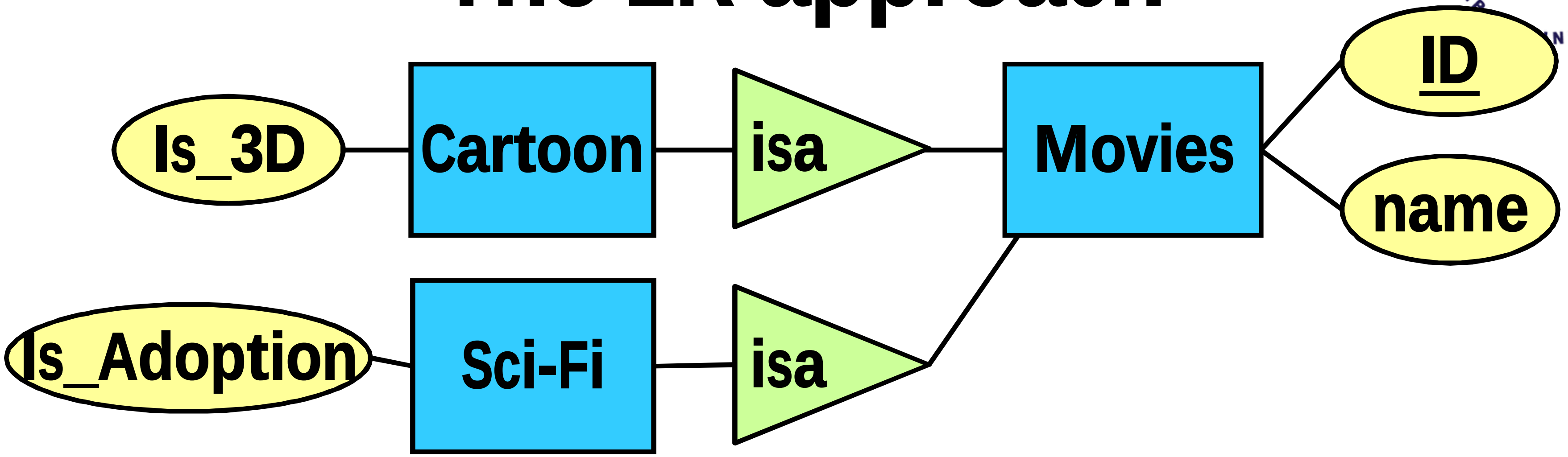
state-name	city-name	population
...	...	...

Subclass → Relation



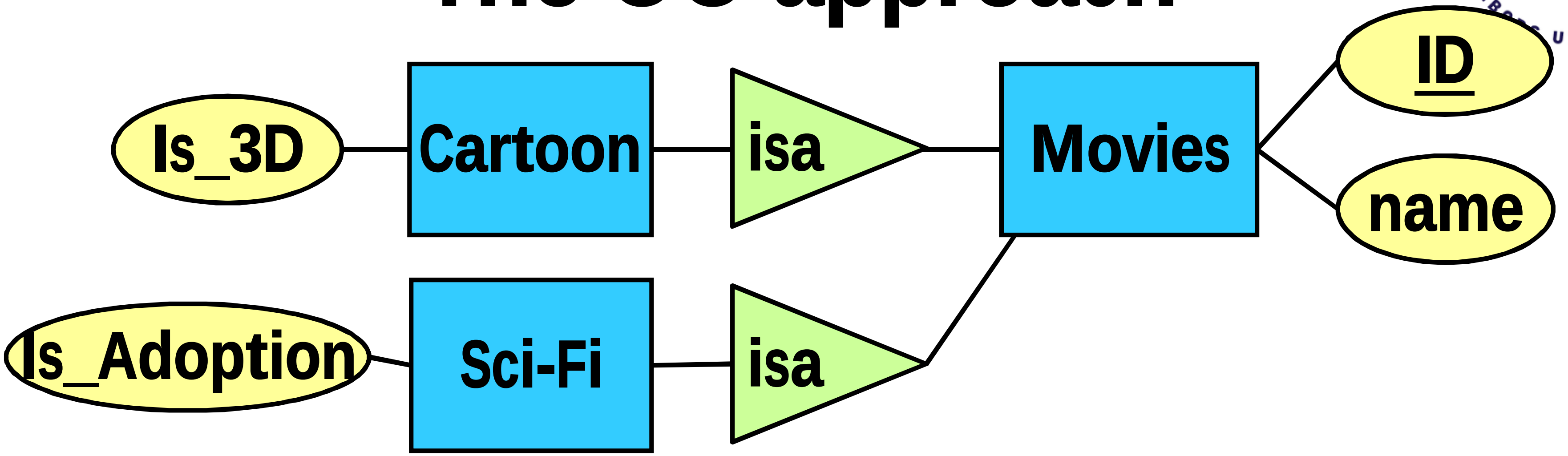
- There are three different ways
  - The ER approach
  - The OO approach
  - The NULL approach

The ER approach



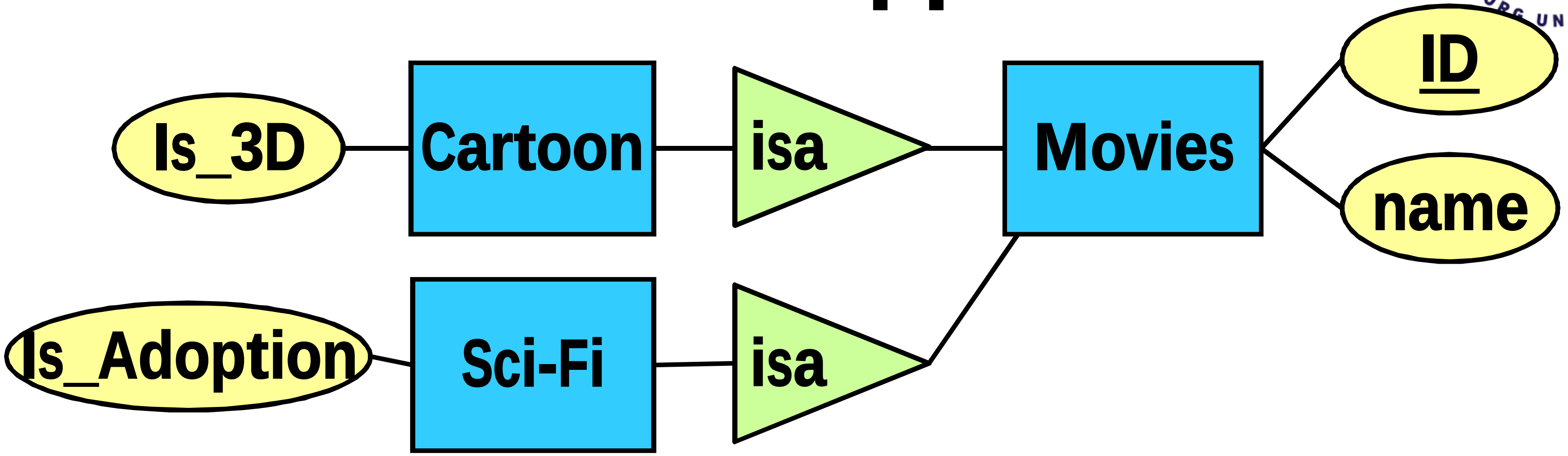
- One relation for each entity set
  - Movies(ID, name)
  - Cartoon(ID, Is_3D)
  - Sci-Fi(ID, Is_Adoption)
- A record may appear in multiple relations

The OO approach



- One relation for each entity set and each possible subclass combination
  - Movies(ID, name)
  - Cartoon(ID, name, Is_3D)
  - Sci-Fi(ID, name, Is_Adoption)
  - Sci-Fi-Cartoon(ID, name, Is_3D, Is_Adoption)
- Each record appears in only one relation

The NULL approach



- One relation that includes everything
  - Movies(<u>ID</u>, name, Is_3D, Is_Adoption)
- For non-cartoon movies, its “Is_3D” is set to NULL
- For non-sci-fi movies, its “Is_Adoption” is set to NULL

Which Approach is the Best?

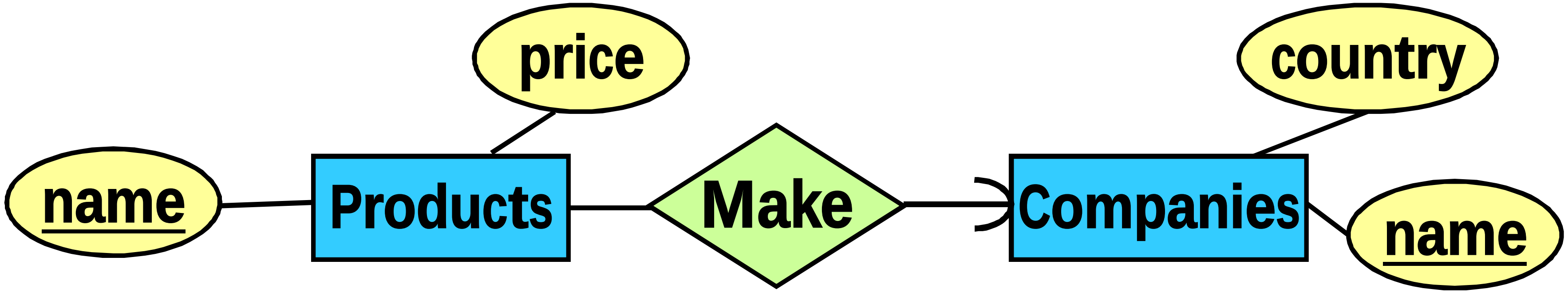
- It depends
- The NULL approach
  - Advantage: Needs only one relation
  - Disadvantage: May have many NULL values
- The OO approach
  - Advantage: Good for searching subclass combinations
  - Disadvantage: May have too many tables
- The ER approach
  - A middle ground between OO and NULL

Questions?

ER Diagram → Relational Schema

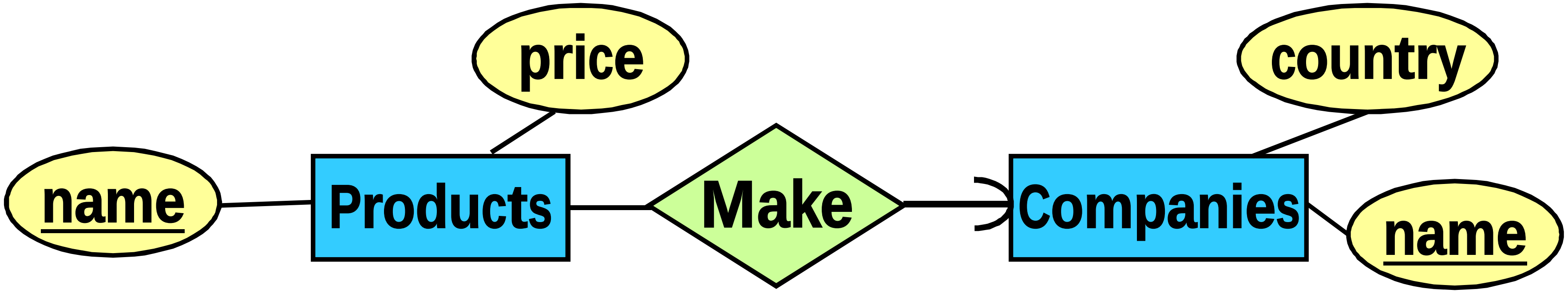
- General rules:
  - Each entity set becomes a relation
  - Each many-to-many relationship becomes a relation
- Special treatment needed for:
  - Weak entity sets
  - Subclasses
  - Many-to-one and one-to-one relationships

Many-to-One Relationship → Relation



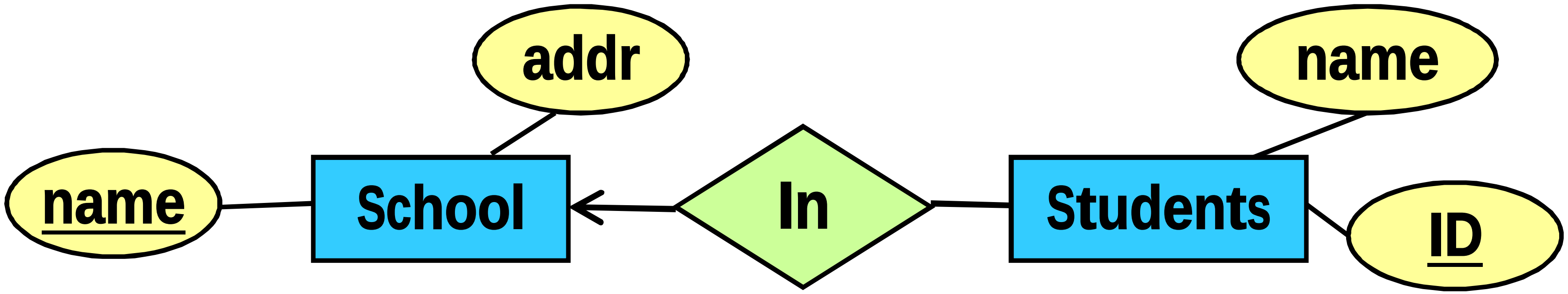
- Intuitive translation:
  - Products(<u>Pname</u>, price)
  - Companies(<u>Cname</u>, country)
  - Make(<u>Pname</u>, Cname)
- Observation: in “Make”, each Pname has only one Cname
- Simplification: Merge “Make” and “Products”
- Results:
  - Products(<u>Pname</u>, price, Cname)
  - Companies(<u>Cname</u>, country)

Many-to-One Relationship → Relation



- In general, we do not need to create a relation for a many-to-one relationship
- Instead, we only need to put the key of the “one” side into the relation of the “many” side

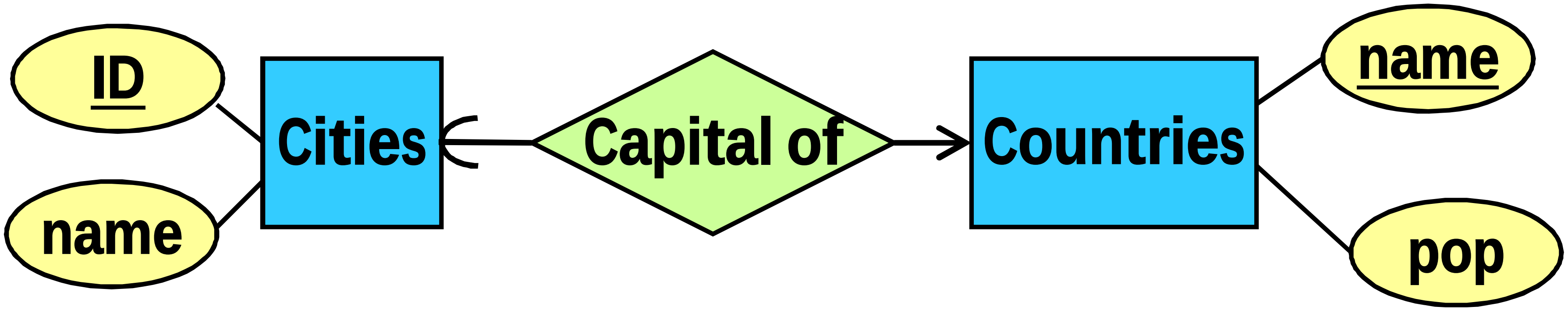
Many-to-One Relationship → Relation



- Translation:
  - School(<u>Hname</u>, addr)
  - Students(<u>ID</u>, Sname, Hname)

- Only need to put the key of the “one” side into the relation of the “many” side

One-to-One Relationship → Relation



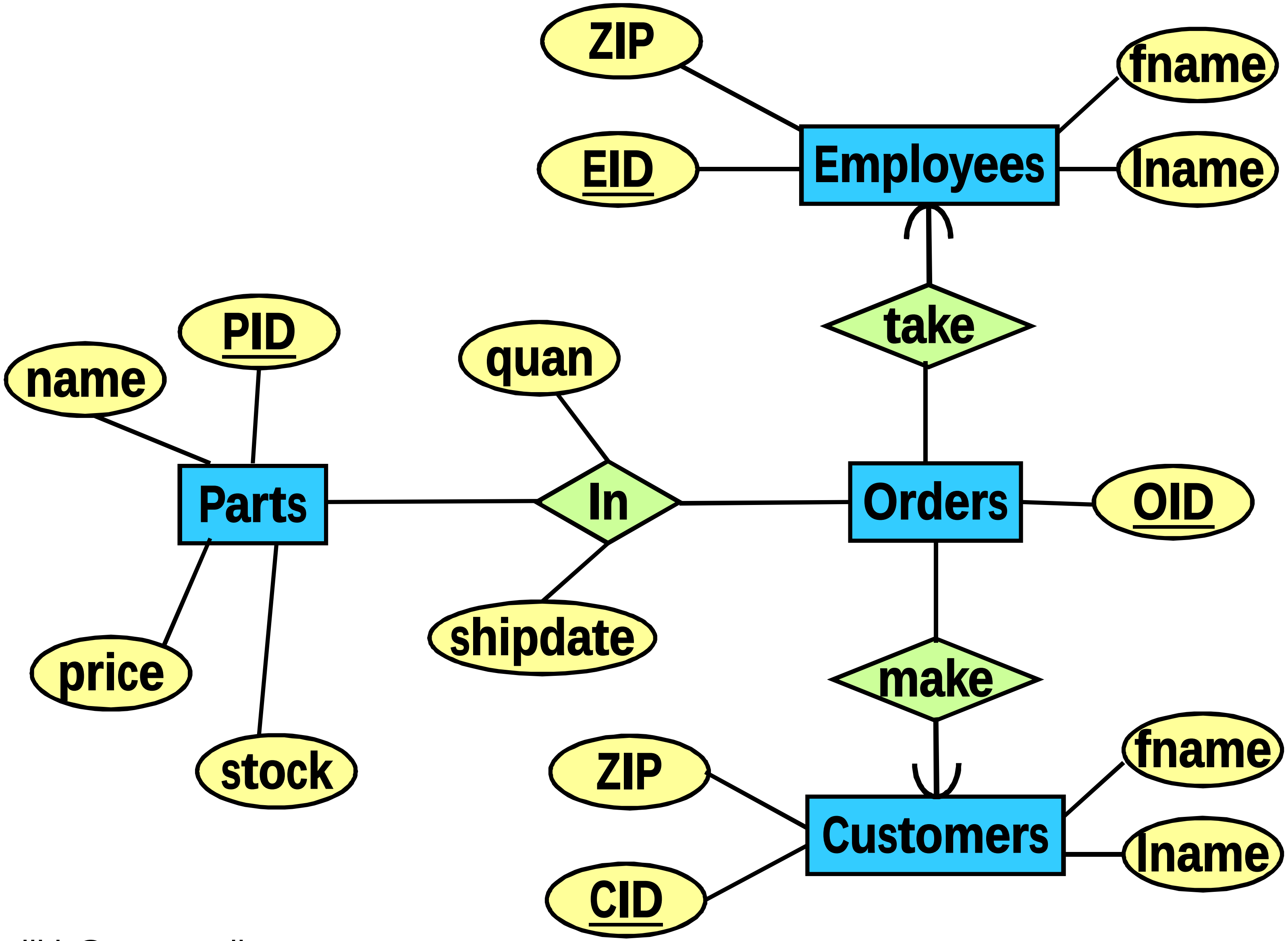
- No need to create a relation for a one-to-one relationship
- Only need to put the key of one side into the relation of the other
- Solution 1
  - Cities(TID, Tname)
  - Countries(Cname, pop, TID)
- Solution 2
  - Cities(TID, Tname, Cname)
  - Countries(Cname, pop)

Questions?

Classroom Exercise-1

- Consider a mail order database in which employees take orders for parts from customers. The requirements are:
- Each employee is identified by a unique employee number, and has a first name, a last name, and a zip code.
- Each customer is identified by a unique customer number, and has a first name, last names, and a zip code.
- Each part being sold is identified by a unique part number. It has a part name, a price, and a quantity in stock.
- Each order placed by a customer is taken by one employee and is given a unique order number. Each order may contain certain quantities of one or more parts. The shipping date of each part is also recorded.

Classroom Exercise-1



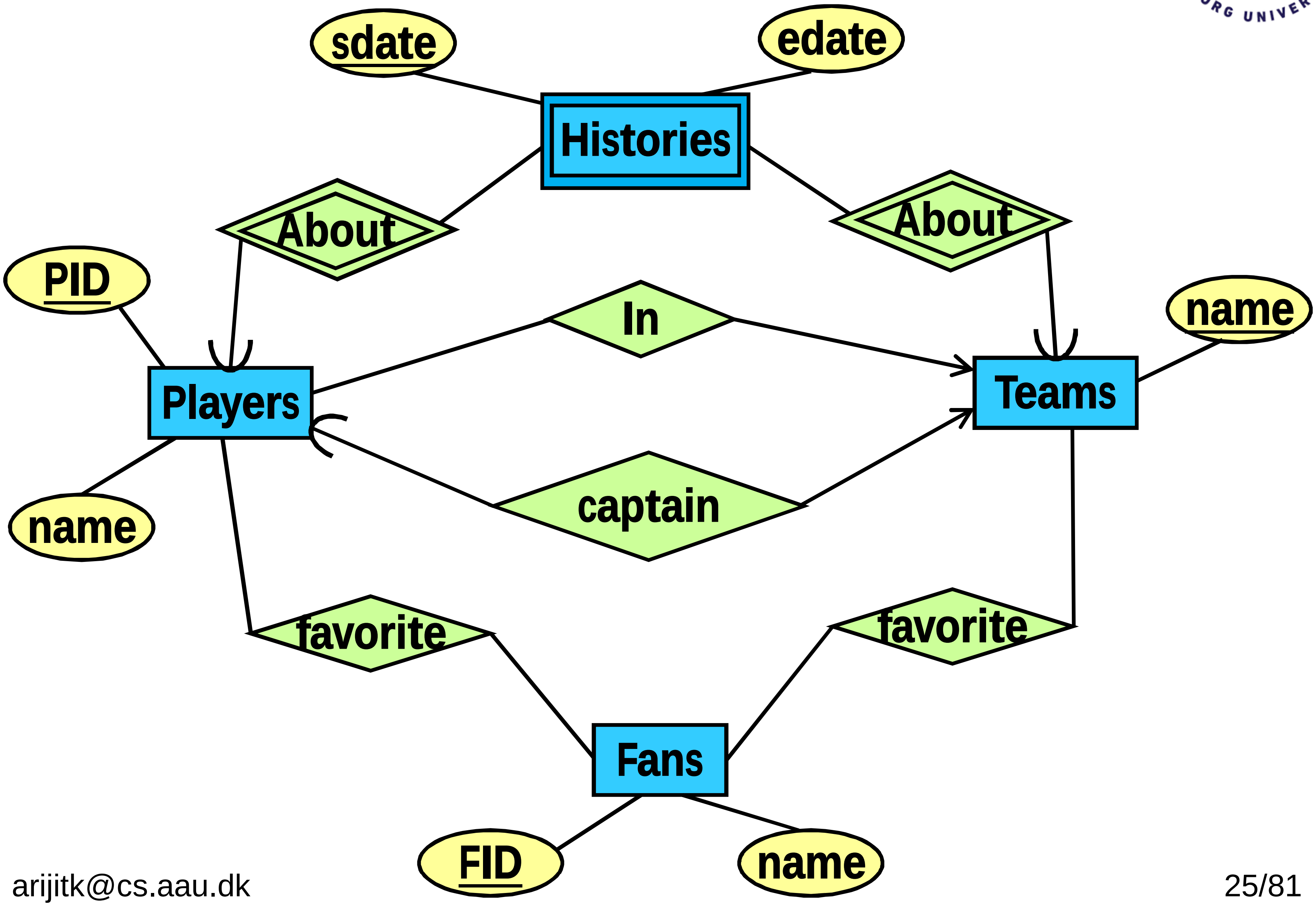
Classroom Exercise-1: Solution

- Parts(<u>PID</u>, name, price, stock)
- Employees(<u>EID</u>, fname, lname, ZIP)
- Customers(<u>EID</u>, fname, lname, ZIP)
- Orders(<u>OID</u>, EID, CID)
- In(<u>OID</u>, <u>PID</u>, quan, shipdate)

Classroom Exercise-2

- Record info about teams, players, and their fans, including:
  - For each team, its name, its players, its team captain (who is also a player)
  - For each player, his/her name, and the history of teams on which he/she has played, including the start and ending dates for each team
  - For each fan, his/her name, favorite teams, favorite players
- Additional information:
  - Each team has at least one player, and exactly one captain
  - Each team has a unique name
  - Two players (or two fans) may have the same name
  - Each fan has at least one favorite team and at least one favorite player

Classroom Exercise-2



Classroom Exercise-2: Solution

- Player(PID, name, team-name)
- Fans(FID, name)
- Teams(team-name, PID)
- Histories(PID, team-name, sdate, edate)
- FavoriteTeam(FID, team-name)
- FavoritePlayer(FID, PID)

5 Minutes Interval

The important thing is not to stop questioning.

Albert Einstein

Database Design

- Applications → ER diagrams → tables
- It works in general, but sometimes things might go wrong
  - The ER diagrams might not be well designed
  - The conversion from ER diagrams to tables might not be done properly
- As a consequence, the resulting tables might have various problems

Data Anomalies

Name	CPR	PhoneNumber	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- Primary key of the table:
  (CPR, PhoneNumber)
- There are several anomalies in the table
- First, redundancy:
  - Alice's address is duplicated

Data Anomalies

Name	<u>CPR</u>	<u>PhoneNumber</u>	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- Primary key of the table:
  (CPR, PhoneNumber)
- Second, update anomalies:
  - We may accidentally update one of Alice's addresses, leaving the other unchanged

Data Anomalies

Name	CPR	PhoneNumber	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- Primary key of the table:
  (CPR, PhoneNumber)
- Third, deletion anomalies:
  - Bob no longer uses a phone
  - Can we remove Bob's phone number?
  - No. (Note: Primary key attributes cannot be NULL)

Data Anomalies

Name	<u>CPR</u>	<u>PhoneNumber</u>	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- Primary key of the table:
  (CPR, PhoneNumber)
- Fourth, insertion anomalies:
  - Name = Cathy, CPR = 9394, HomeAddress = YiShun
  - Can we insert this information into the table?
  - No. (Note: Primary key attributes cannot be NULL)

Normalization

Name	CPR	PhoneNumber	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- How do we get rid of those anomalies?
- Normalize the table (i.e., decompose it)

Name	CPR	HomeAddress
Alice	1234	Jurong East
Bob	5678	Pasir Ris

CPR	PhoneNumber
1234	67899876
1234	83848384
5678	98765432

Effects of Normalization

Name	CPR	HomeAddress
Alice	1234	Jurong East
Bob	5678	Pasir Ris

CPR	PhoneNumber
1234	67899876
1234	83848384
5678	98765432

- Duplication?
  - No. (Alice's address is no longer duplicated.)
- Update anomalies?
  - No. (There is only one place where we can update the address of Alice)
- Deletion anomalies?
  - No. (We can freely delete Bob's phone number)
- Insertion anomalies?
  - No. (We can insert an individual with a phone)

Questions?

Functional Dependencies: Intuition

Name	CPR	PhoneNumber	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- Why was this table bad?
- It has a lot of anomalies
- Why does it have those anomalies?
- Intuitive answer: It contains a bad combination of attributes

Functional Dependencies: Intuition

Name	CPR	PhoneNumber	HomeAddress
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- In general, how do we know whether a combination of attributes is bad?
- We need to check the correlations among those attributes
- What kind of correlations?
- Functional dependencies (FD)

Functional Dependencies (FD)

- Consider two attributes in practice: CPR, Name
- Given an CPR, can we always uniquely identify the name of the person?
- Theoretically yes -- We just need help from Govt.
- Given a Name, can we always uniquely identify the CPR of the person
- In general no -- Different people can have the same name
- Therefore
  - CPR decides Name, but not vice versa
  - Functional dependencies:
    CPR $\rightarrow$ Name, but not Name $\rightarrow$ CPR

Functional Dependencies (FD)

- Consider three attributes in practice:
  - StudentID, Name, Address, PostalCode
- Functional Dependencies:
  - StudentID → Name, Address, PostalCode
  - Address → PostalCode
  - PostalCode → Address

Classroom Exercise-3

Name	Category	Color	Department	Price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Black	Toys	99
Gizmo	Stationary	Green	Office Supplies	59

- Find the functional dependencies are definitely not true on the above table
  - Category → Department
  - Category, Color → Price
  - Price → Color
  - Name → Color
  - Department, Category → Name
  - Color, Department → Name, Price, Category

Classroom Exercise-3: Solution

Name	Category	Color	Department	Price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Black	Toys	99
Gizmo	Stationary	Green	Office Supplies	59

- Find the functional dependencies are definitely not true on the above table
  - Category → Department
  - Category, Color → Price
  - Price → Color
  - **Name → Color** ❌
  - **Department, Category → Name** ❌
  - Color, Department → Name, Price, Category

Where Do FDs Come From?

- From common sense
- From the application's requirements
- Example
  - Purchase(CustomerID, ProductID, ShopID, Price, Date)
  - Requirement: Each shop can sell at most one product
  - FD implied: ShopID → ProductID

Where Do FDs Come From?

- Example
  - Purchase(CustomerID, ProductID, ShopID, Price, Date)
  - Requirement: No two customers buy the same product
  - FD implied: ProductID → CustomerID

Where Do FDs Come From?

- Example
  - Purchase(CustomerID, ProductID, ShopID, Price, Date)
  - Requirement: No shop will sell the same product to the same customer on the same date at two different prices
  - FD implied:
    CustomerID, ProductID, ShopID, Date → Price

Reasoning with FDs

- Example:
  - Given:
    CPR → Address,   Address → PostalCode
  - We have: CPR → PostalCode
- In general
  - Given A→B, B→C
  - We always have A→C
- Any others rules? Armstrong's Axioms

Armstrong's Axioms

- Axiom of Relexivity
  - A set of attributes → A subset of the attributes
- Example
  - CPR, Name → CPR
  - StudentID, Name, Age → Name, Age
  - ABCD → ABC
  - ABCD → BCD
  - ABCD → AD

Armstrong's Axioms

- Axiom of Augmentation
  - Given A → B
  - We always have AC → BC, for any C
- Example
  - If CPR → Name
  - Then CPR, Age → Name, Age
  - and CPR, Salary, Weight → Name, Salary, Weight
  - and CPR, Addr, Postal → Name, Addr, Postal

Armstrong’s Axioms

- Axiom of Transitivity
  - Given A → B and B → C
  - We always have A → C
- Example
  - If CPR → Addr, and Addr → Postal
  - Then CPR → Postal

Reasoning with FDs

- Given $A \rightarrow B$, $BC \rightarrow D$
- Can you prove that $AC \rightarrow D$?
- Proof
  - Given $A \rightarrow B$, we have $AC \rightarrow BC$ (Augmentation)
  - Given $AC \rightarrow BC$ and $BC \rightarrow D$, we have $AC \rightarrow D$ (Transitivity)

Reasoning with FDs

- Given A→B, D→C
- Can you prove that AD → BC?
- Proof
  - Given A→B, we have AD → BD (Augmentation)
  - Given AD→BD, we have AD → B (Reflexivity)
  - Given D→C, we have AD → AC (Augmentation)
  - Given AD → AC, we have AD → C (Reflexivity)
  - In other words, AD decides B and C
  - Therefore, AD → BC

Reasoning with FDs

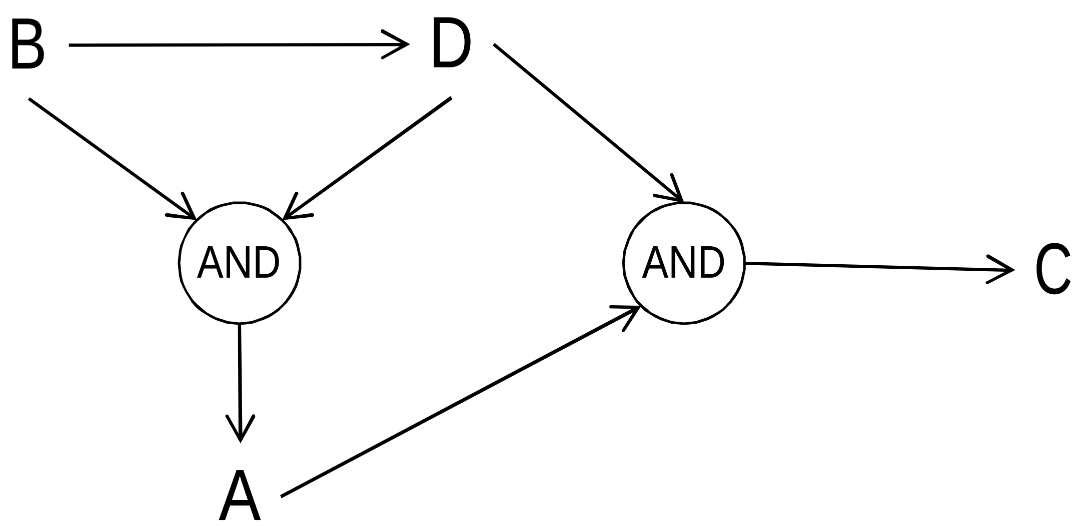
- Given A→C, AC→D, AD→B
- Can you prove that A → B?
- Proof
  - Given A → C, we have A → AC (Augmentation)
  - Given A → AC and AC → D, we have A → D (Transitivity)
  - Given A → D, we have A → AD (Augmentation)
  - Given A → AD and AD → B, we have A → B (Transitivity)

Reasoning with FDs

- Four attributes: A, B, C, D
- Given: $B \rightarrow D$, $DB \rightarrow A$, $AD \rightarrow C$
- Can you prove $B \rightarrow C$?
- Doable with Armstrong's axioms, but troublesome
- We will discuss a more convenient approach

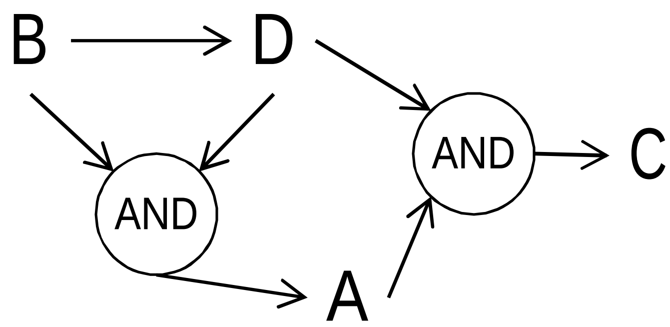
An Intuitive Solution

- Four attributes: A, B, C, D
- Given: B→D, DB→A, AD→C
- Can you prove B→C?



Steps of the Intuitive Solution

- Four attributes: A, B, C, D
- Given: B→D, DB→A, AD→C
- Can you prove B→C?



- First, activate B
  - Activated set = { B }
- Second, activate whatever B can activate
  - Activated set = { B, D }, since B→D
- Third, use all activated elements to activate more
  - Activated set = { B, D, A }, since DB→A
- Repeat the third step, until no more activation is possible
  - Activated set = { B, D, A, C }, since AD→C; done

Questions?

Classroom Exercise-4

- Given: $A \rightarrow C$, $C \rightarrow B$, $B \rightarrow D$, $D \rightarrow E$, $E \rightarrow A$
- Can you prove $C \rightarrow ABE$?
- We start with {C}
- Since $C \rightarrow B$, we have {C, B}
- Since $B \rightarrow D$, we have {C, B, D}
- Since $D \rightarrow E$, we have {C, B, D, E}
- Since $E \rightarrow A$, we have {C, B, D, E, A}
- A, B, E are all in the set, so $C \rightarrow ABE$ holds

Classroom Exercise-5

- Given: C→D, AD→E, BC→E, E→A, D→B
- Can you prove C→A?
- We start with {C}
- Since C→D, we have {C, D}
- Since D→B, we have {C, D, B}
- Since BC→E, we have {C, D, B, E}
- Since E→A, we have {C, D, B, E, A}
- A is in the set, so C→A holds

Classroom Exercise-6

- Given: $C \rightarrow D$, $AD \rightarrow E$, $BC \rightarrow E$, $E \rightarrow A$, $D \rightarrow B$, $B \rightarrow F$
- Can you prove $D \rightarrow C$?
- We start with {D}
- Since $D \rightarrow B$, we have {D, B}
- Since $B \rightarrow F$, we have {D, B, F}
- What else?
- No more.
- {D, B, F} is all what can be decided by D
- We refer to {D, B, F} as the closure of D

Classroom Exercise-7

- A table with five attributes A, B, C, D, E, F
- $AB \rightarrow C$, $AD \rightarrow E$, $B \rightarrow D$, $AF \rightarrow B$
- Compute the following closures
  - $\{B, C\}^+ =$
  - $\{A, B\}^+ =$
  - $\{A, F\}^+ =$

Classroom Exercise-7: Solution

- A table with five attributes A, B, C, D, E, F
- $AB \rightarrow C$, $AD \rightarrow E$, $B \rightarrow D$, $AF \rightarrow B$
- Compute the following closures
  - $\{B, C\}^+ = \{B, C, D\}$
  - $\{A, B\}^+ = \{A, B, C, D, E\}$
  - $\{A, F\}^+ = \{A, F, B, C, D, E\}$

Closure & FD

- To prove that X → Y holds, we only need to show that $\{X\}^+$ contains Y
- AB→C, AD→E, B→D, AF→B
- Prove that AF→D
- $\{AF\}^+ = \{AFBCDE\}$, which contains D
- Therefore, AF→D holds

Closure & FD

- To prove that X → Y does not hold, we only need to show that $\{X\}^+$ does not contain Y
- AB→C, AD→E, B→D, AF→B
- Prove that AD→F does not hold
- $\{AD\}^+ = \{ADE\}$, which does not contain F
- Therefore, AF→F does not hold

Questions?

Superkeys of a Table

Name	CPR	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

- Definition: A set of attributes in a table that decides all other attributes
- Example:
  - {CPR} is a superkey
  - Since CPR → Name, Postal, Address
  - {CPR, Name} is a superkey
  - Since {CPR, Name} → Postal, Address

Keys of a Table

Name	CPR	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

- Definition: A superkey that is minimal
- i.e., if we remove any attribute from the superkey, it will not be a superkey anymore
- Example:
  - {CPR} is a superkey
  - Since CPR → Name, Postal, Address
  - {CPR, Name} is a superkey
  - Since {CPR, Name} → Postal, Address
  - CPR is a key, but {CPR, Name} is not a key

Keys of a Table

Name	<u>CPR</u>	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

- Note: Not to be confused with the keys of entity sets

Candidate Keys

Name	CPR	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3576	3	420923	Yishun

- A table may have multiple keys
- In that case, each key is referred to as a candidate key
- Example:
  - {CPR} is a key
  - Since CPR → Name, StudentID, Postal, Address
  - {StudentID} is a key
  - Since StudentID → Name, CPR, Postal, Address
  - Both {CPR} and {StudentID} are candidate keys

Primary and Secondary Keys

Name	CPR	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3576	3	420923	Yishun

- When a table have multiple keys…
- We choose one of them as the primary key
- The others are referred to as secondary keys
- Example:
  - {CPR} is a key
  - {StudentID} is a key
  - If we choose {CPR} as the primary key
  - Then {StudentID} is the secondary key

Summary

- **ER Diagram → Relational Schema** ✓
- **Functional Dependency (FD)** ✓
- **Keys** ✓

Study-at-Home Slides

Finding the (Candidate) Keys

Will be in the syllabus of Exercise, self-study, and Exam

Finding the (Candidate) Keys

- To check whether a table is “good”, we need to find the keys of the table
- How do we do that?
- Use functional dependencies (FDs) and closures

Example

- Definition of a Key: A minimal set of attributes that decides all other attributes
- A table R(A, B, C)
- FDs given: $A \rightarrow B$, $B \rightarrow C$
- Is A a key?
- $\{A\}^+ = \{ABC\}$, i.e., $A \rightarrow ABC$. Yes.
- Is B a key?
- $\{B\}^+ = \{BC\}$, i.e., B does not decide A. No.
- Is C a key?
- $\{C\}^+ = \{C\}$. No.
- Is AB a key?
- No, since A is already a key.
- What about BC, AC, ABC?

Example

- Definition of a Key: A minimal set of attributes that decides all other attributes
- A table R(A, B, C)
- FDs given: $A \rightarrow B$
- Is A a key?
- $\{A\}^+ = \{A, B\}$. No.
- Is B or C a key?
- $\{B\}^+ = \{B\}$. $\{C\}^+ = \{C\}$. No.
- Is AB or BC a key?
- $\{AB\}^+ = \{AB\}$. $\{BC\}^+ = \{BC\}$. No.
- Is AC a key?
- $\{AC\}^+ = \{ABC\}$. Yes.
- Is ABC a key?

Finding the Keys: Algorithm

- Check all possible combinations of attributes in the table
  - Example: A, B, C, AB, BC, AC, ABC
- For each combination, compute its closure
  - Example: $\{A\}^+ = ...$, $\{B\}^+ = ...$, $\{C\}^+ = ...$, ...
- If a closure contains all attributes, then the combination might be a key (or superkey)
  - Example: $\{A\}^+ = \{ABC\}$
- Make sure that you select only keys
  - Example: $\{A\}^+ = \{ABC\}$, $\{AB\}^+ = \{ABC\}$, don't select AB

Example

- A table R(A, B, C, D)
- AB→C, AD→B, B→D
- First, enumerate all attribute combinations:
  - {A}, {B}, {C}, {D}
  - {AB}, {AC}, {AD}, {BC}, {BD}, {CD}
  - {ABC}, {ABD}, {ACD}, {BCD}
  - {ABCD}

Example

- A table R(A, B, C, D)
- AB→C, AD→B, B→D
- Second, compute the closures:
  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{BD\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
  - $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{AC\}$, $\{AD\}^+ = \{ABCD\}$
  - $\{BC\}^+ = \{BCD\}$, $\{BD\}^+ = \{BD\}$, $\{CD\}^+ = \{CD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{ACD\}^+ = \{ABCD\}$
  - $\{BCD\}^+ = \{BCD\}$
  - $\{ABCD\}^+ = \{ABCD\}$
- Finally, output the keys

A Small Trick

- Always check small combinations first
- A table R(A, B, C, D)
- $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$
- Compute the closures:
  - $\{A\}^+ = \{ABCD\}$, $\{B\}^+ = \{ABCD\}$, $\{C\}^+ = \{ABCD\}$, $\{D\}^+ = \{ABCD\}$
  - No need to check others
  - The others are all superkeys but not keys
- Keys: {A}, {B}, {C}, {D}

Another Small Trick

- A table R(A, B, C, D)
- AB→C, AD→B, B→D
- Notice that A does not appear in the right hand side of any functional dependencies
- In that case, A must be in every key
- Keys of R: AB, AD (From the previous slide)
- In general, if an attribute that does not appear in the right hand side of any FD, then it must be in every key

Exercise (Find the Keys)

- A table R(A, B, C, D)
- A→B, A→C, C→D
- A must be in every key
- Compute the closures:
  - $\{A\}^+ = \{ABCD\}$
  - No need to check others
- Keys: {A}

Exercise (Find the Keys)

- A table R(A, B, C, D, E)
- $AB \rightarrow C$, $C \rightarrow B$, $BC \rightarrow D$, $CD \rightarrow E$
- A must be in every key
- Compute the closures:
  - $\{A\}^+ = \{A\}$
  - $\{AB\}^+ = \{ABCDE\}$
  - $\{AC\}^+ = \{ACBDE\}$
  - $\{AD\}^+ = \{AD\}$, $\{AE\}^+ = \{AE\}$
  - $\{ADE\}^+ = \{ADE\}$

Exercise (Find the Keys)

- A table R(A, B, C, D, E, F)
- AB→C, C→B, CBE→D, D→EF
- A must be in every key
- Compute the closures:
  - $\{A\}^+ = \{A\}$
  - $\{AB\}^+ = \{ABC\}$
  - $\{AC\}^+ = \{ACB\}$
  - $\{AD\}^+ = \{ADEF\}$
  - $\{AE\}^+ = \{AE\}$, $\{AF\}^+ = \{AF\}$
  - $\{ABC\}^+ = \{ABC\}$
  - $\{ABD\}^+ = \{ABE\}^+ = \{ACD\}^+ = \{ACE\}^+ = \{ABCDEF\}$
  - $\{ADE\}^+ = \{ADEF\}$

Questions ??

Thank You !