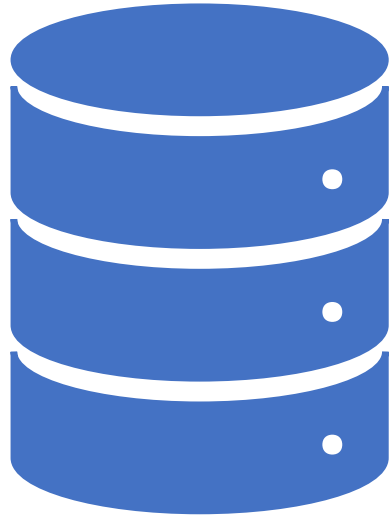


Advanced Databases

Relational Design Revisited





Why are
relational
databases/DBMS
useful?

Why are databases/DBMS useful?

Storage

- Store large volumes of data efficiently

Access/ Processing

- Multiple users can read and modify data at the same time
- Can search, sort to suit different access requirements
- Data sharing across applications

Management

- Easy to add/update/delete data
- Can extend/amend structures to suit organisational requirements

Security

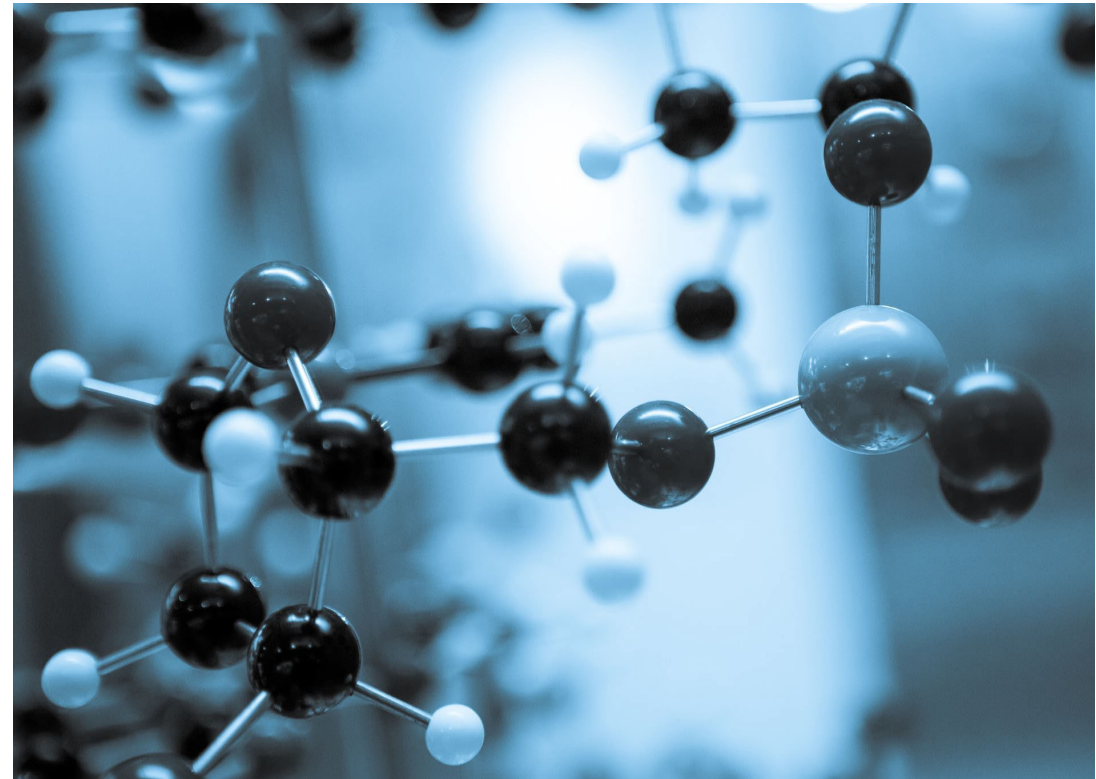
- Mechanisms for backup, distribution, redundancy
- Restricting access
- Insulating data from programs

Integrity

- ACID
- Can ensure data is correct or has integrity

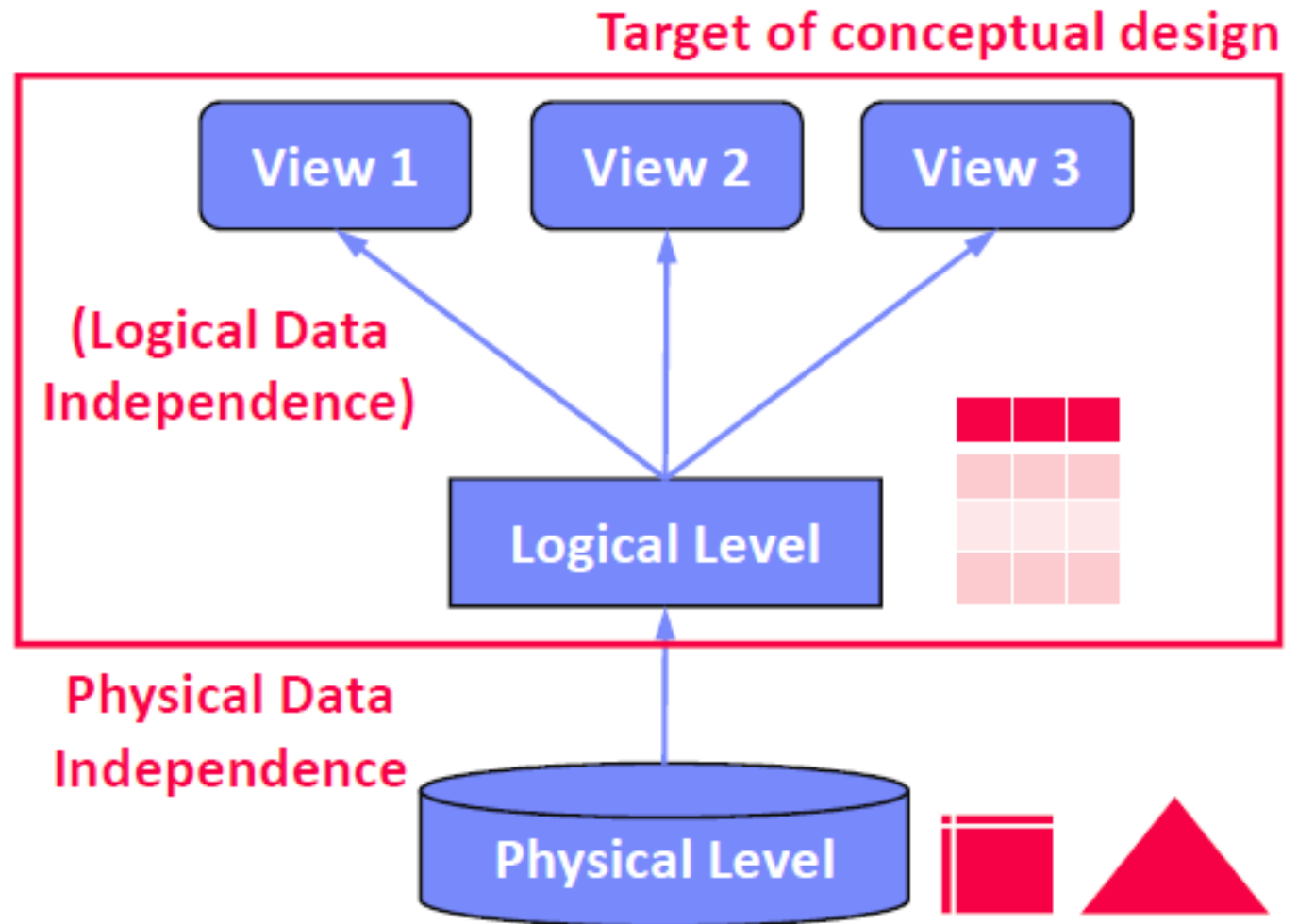
Integrity

- **Atomicity**
 - When changing data, if any part of the change fails, the whole change will fail.
 - Therefore the data will remain as it was before the change was attempted
 - Prevents partial records being created
- **Consistency:**
 - Before data can be changed in a database, it must be validated against a set of rules
- **Isolation:**
 - Multiple changes are possible at the same time, but each change is isolated from others
- **Durability:**
 - Once a change has been made, the data is safe, even in the event of system failure



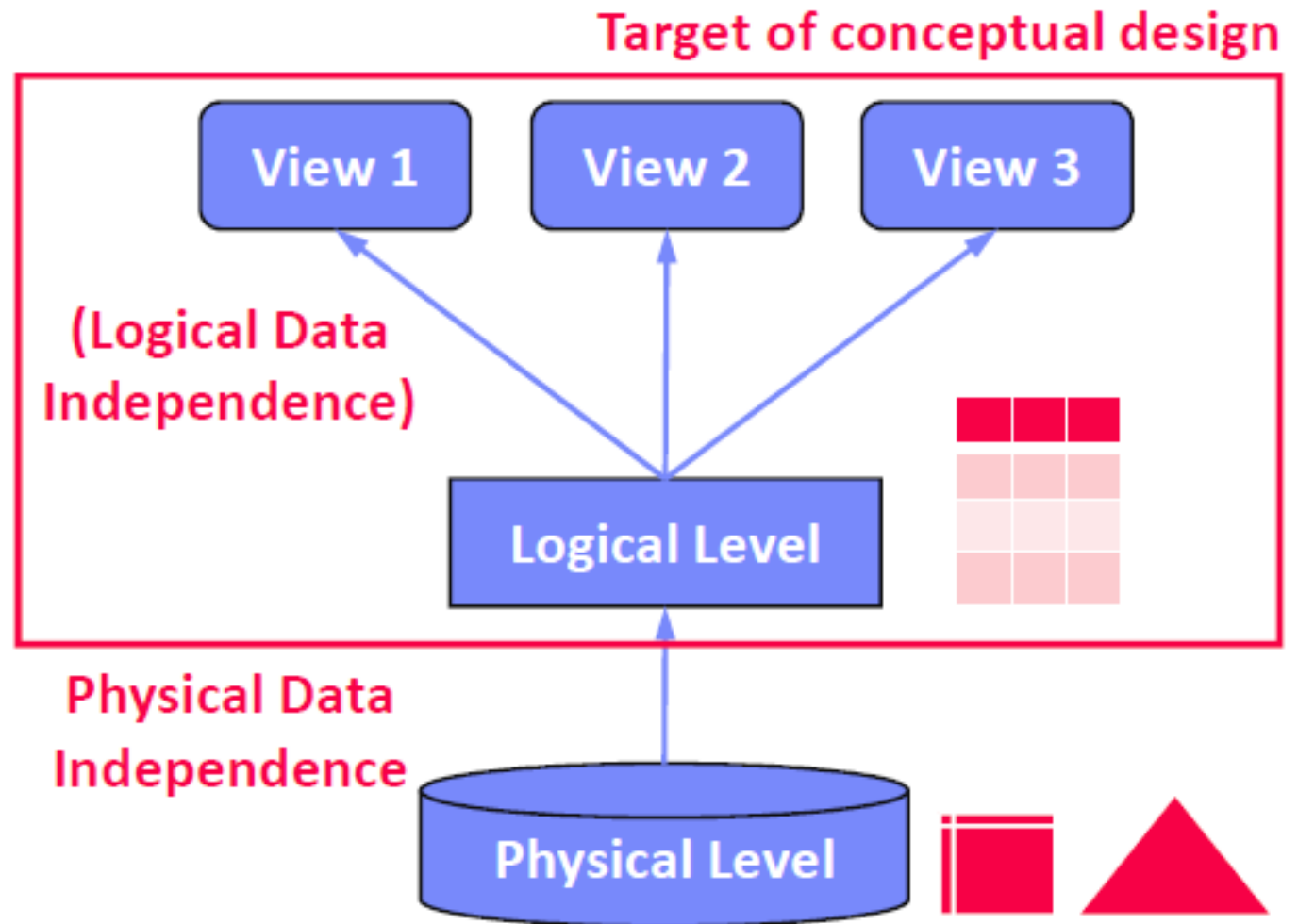
Quick Recap

- Three Layer ANSI-SPARC Architecture
 - 3 levels of abstraction
 - Aims to separate each user's view of the database from the way the database is physically represented
- External schemas
 - (external level)
- Conceptual schema
 - (logical level)
- Internal schema
 - (physical level)

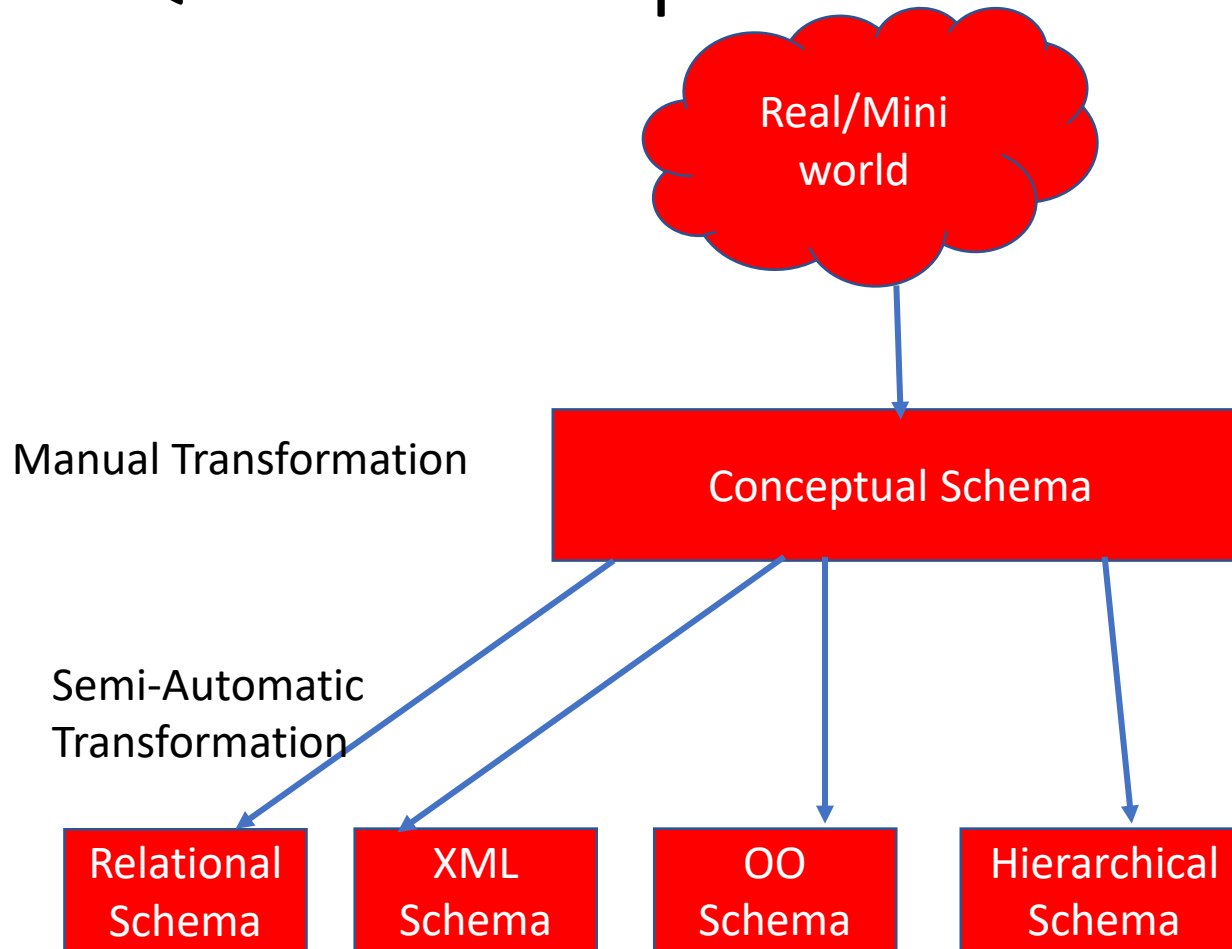


Quick Recap

- Types of Data Independence
- Logical data independence
 - (external views and applications independent of logical data model)
- Physical data independence
 - (logical data model independent of underlying data organization)



Quick Recap



- **Data Model:**

- Concepts for describing data objects and their relationships (meta model)

- **Schema:**

- Description (structure, semantics) of specific data collection

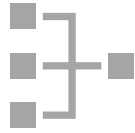
Quick Recap

DB Design



Requirements engineering

Collect and analyze data and application requirements
Specification documents



Conceptual Design

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



Logical Design

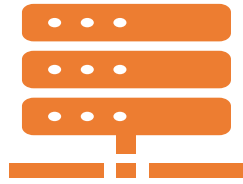
Model data with implementation primitives of concrete data model
e.g., relational schema + integrity constraints, views, permissions, etc



Physical Design

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

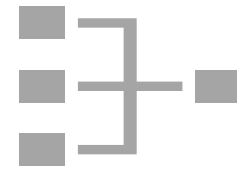
Quick Recap



Conceptual Data Models

Entity-Relationship Model (ERM), focus on data,
~1975

Unified Modeling Language (UML), focus on data
and behaviour, ~1990



Logical Data Models

Relational

Key-Value

Graph

Document (XML, JSON)

Matrix/Tensor

Object-oriented


Network

Hierarchical

ER Model

- Mini World
- Model
 - **Concepts** in this mini world
 - Identifying key characteristics of these concepts - **attributes**
 - **Relationships** between these concepts in this mini world





ER Model – Entity Types

- An object in the mini world about which information is to be stored.
 - Examples: persons, books, courses.
- Entities do not have to correspond to objects of physical existence.
- Entities may also represent conceptual objects e.g., holidays, sale, purchase
 - These are technically Entity types
 - Entities are populated instances of these entity types

ER Model - Entities

- The mini world you model can contain only a finite number of objects.
- It must be possible to distinguish entities from each other
 - i.e. objects must have some **identity**.



ER Model - Attributes

- Properties or feature of an entity type (or relationship).
 - These get populated for each entity
 - Example: the title of a book, name of an author
- The value of an attribute is an element of a data type like string, integer, date.



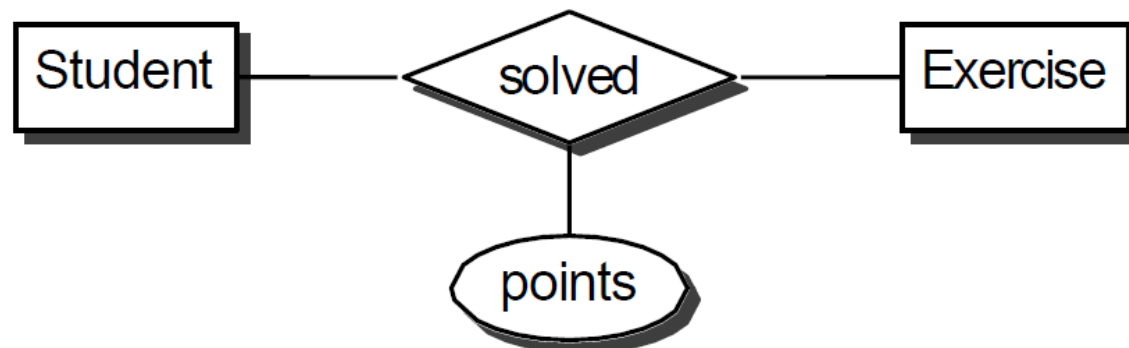
ER Model – Relations

- A relation—occurs between pairs of entities
 - Binary relationship
- Example:
 - Fred (an entity)
 - teaches(a relationship)
 - the module “Advanced Databases” (an entity)



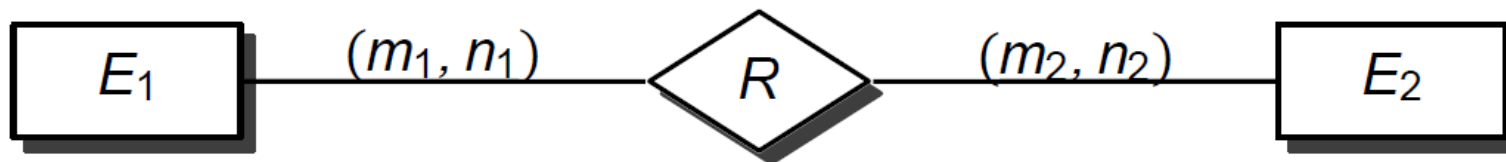
ER Model – Relations

- Relationships may have attributes too
- E.g. This diagram models the fact that a number of points is stored for every pair of a student X and an exercise Y where X submitted a solution or Y.



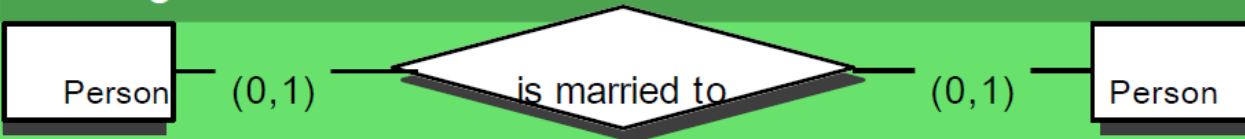
ER Model – Relations Cardinalities

- In general, there is no restriction on how often an entity participates in a relationship R .
- An entity can be connected to one entity of the other type, to more than one
- Specific application semantics dictate how many E_2 entities an E_1 entity can be related to
- The ER model introduces the (\min, \max) notation to specify an interval of possible participations in an relationship



ER Model – Relations Cardinalities

Marriage



"A person can be married to at most one other person and vice versa (at any given point in time)."

Airport Locations

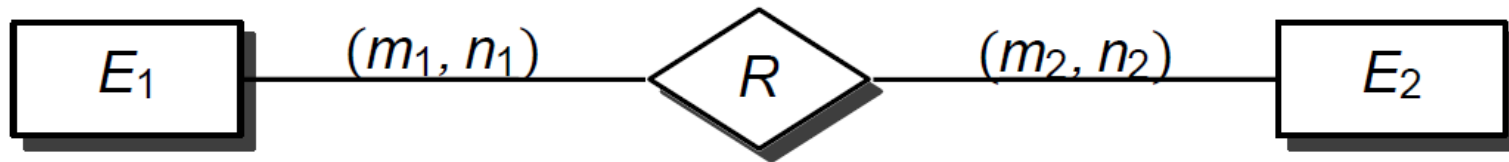


"An airport lies in exactly one country. A country may have arbitrarily many airports (and maybe none at all)."



ER Model – Relations Cardinalities

- To understand a relationship, one must know the cardinality specifications on both sides.
- The maximum cardinalities on each side are used to distinguish between many-to-many, one-to-many / many-to-one, and one-to-one relationships.

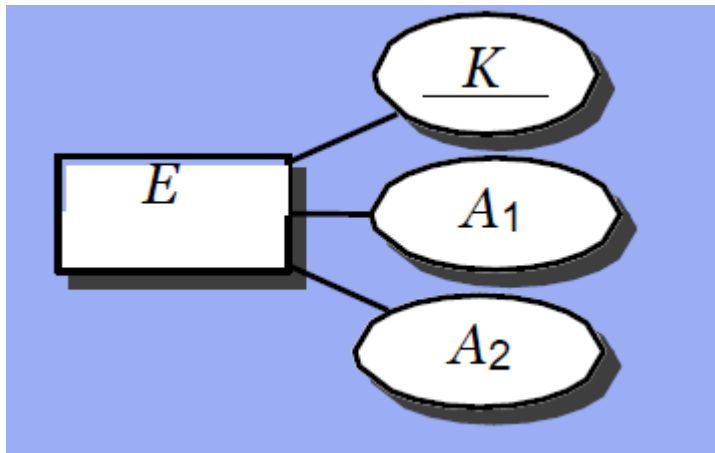


Exercise

- Information about researchers in the data science is to be stored.
- For each researcher, his/her lastname, firstname, e-mail address, and ORCID (a persistent digital identifier) are relevant.
- Researchers are affiliated with universities and assume a certain position(e.g. professor, lecturer).
- Researchers can be affiliated with many universities and may hold different positions in each
- Relevant university information are the name, homepageURI, and country.
- Researchers publish articles in journals.
 - Journals have a title, month and year of publication, volume and issue number
- **Figure out the entities, attributes and relationships (and relationship attributes)**

ER Model - Keys

- A key K of an entity type E is an attribute of E which uniquely identifies the entities of this type.
- No two different entities share the same value for the key attribute.
- Composite keys are allowed.



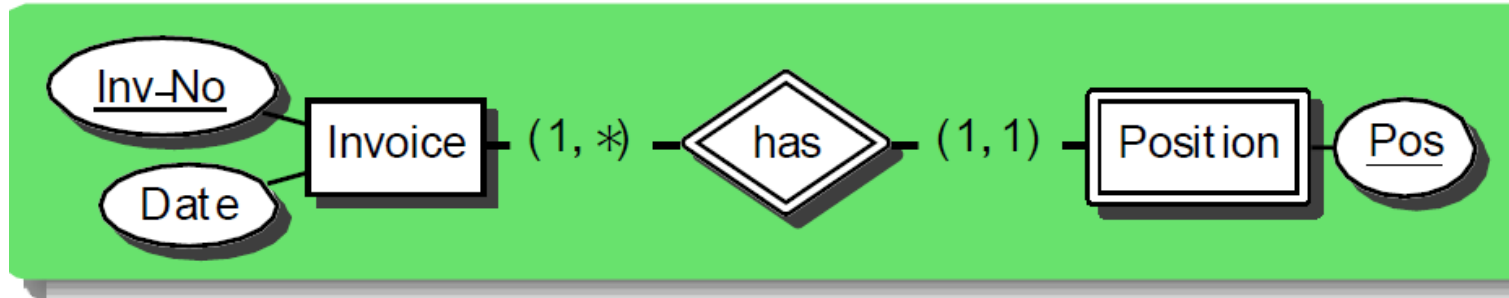
ER Model - Keys

- The translation of ER schemas into relational schemas requires the declaration of ER keys.
- If there is no natural key, add artificial identifiers as attributes(e.g. integers, EMPNO, DEPTNO) which then represent the entities



Weak Entity

- Some entities describe a kind of detail that cannot exist without a master(or owner) entity
- There is a 1:1 cardinality on the detail side to the master
- Plus
- The key of the master is inherited to become part of the detail side



Weak Entity

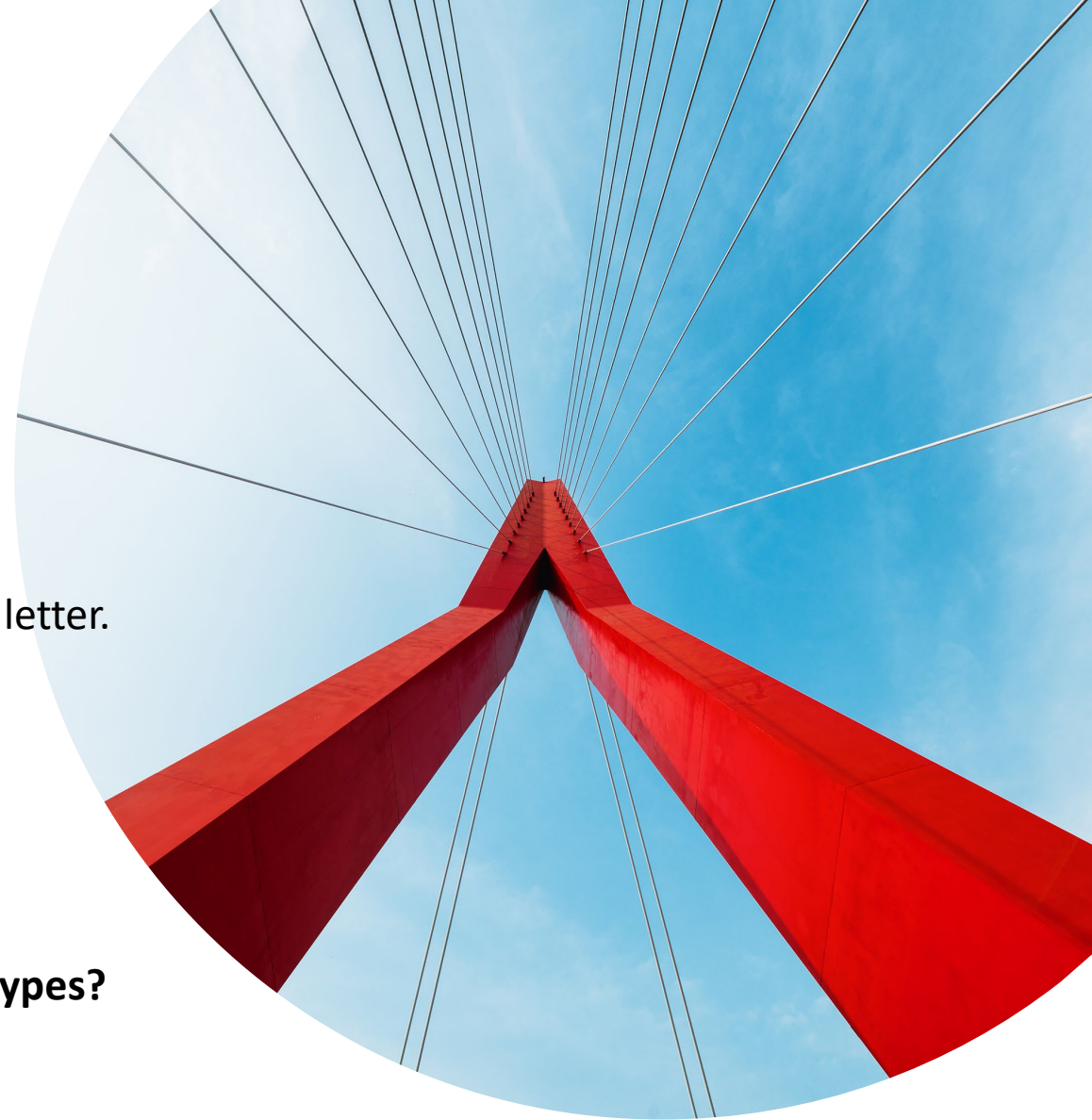
- Examples:
- A classroom is identified by a building and a room number.
- A section in a book is identified by a chapter and a section title.
- A web page URI is composed of a web server DNS address and a path on that server.
- There is also an existence dependency.
 - If the building is pulled down, the classrooms automatically disappear.
- If the web server is shut down, all URIs on that server cease to function.



Weak Entity

- Model a set of online quizzes(multiple choice tests).
- Each quiz is identified by a title
- Each question within a quiz is numbered
- Each possible answer to a given question is referenced by a letter.
- For each question and answer, the associated text is stored.
- Answers are classified into correct and incorrect ones.

- **What is the complete key for each of the occurring entity types?**



A black and white photograph of a baseball field, viewed from an elevated angle, showing the bases and pitcher's mound. The image is slightly blurred and has a high-contrast, artistic feel.

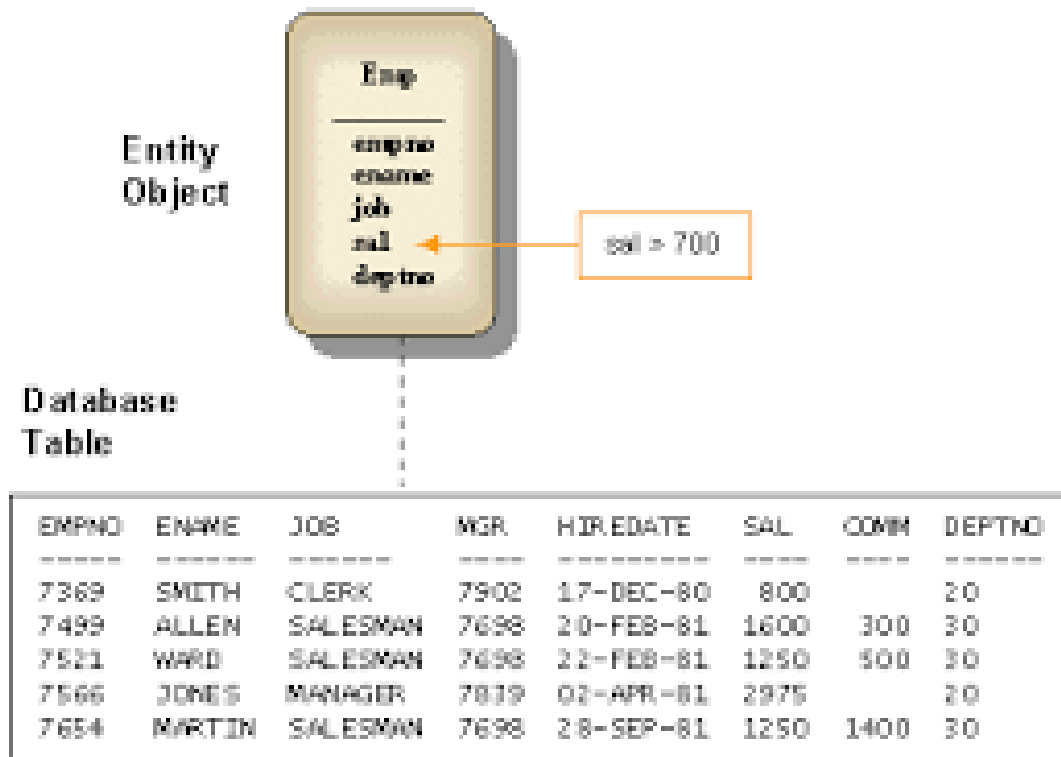
Transforming ER to Relational

Entity v Relation v Table

- Entity:
 - During the very first step in the design process, when you're creating a conceptual design you have a bunch of entities and relationships that represent the various types of data you'll want to store (and express it as an ERD).
- Relation:
 - Once you've finalized your conceptual design you'll convert that ERD into a logical schema (and express it as an ERD).
 - This schema will be a list of relations.
 - The relations are all your entities and relationships from the previous step .
- Table:
 - The final step is to actually create the database with all of its tables based on your schema from the previous step (which was based on your ERD from the first step).
 - The tables are fully defined and usable objects in your database.

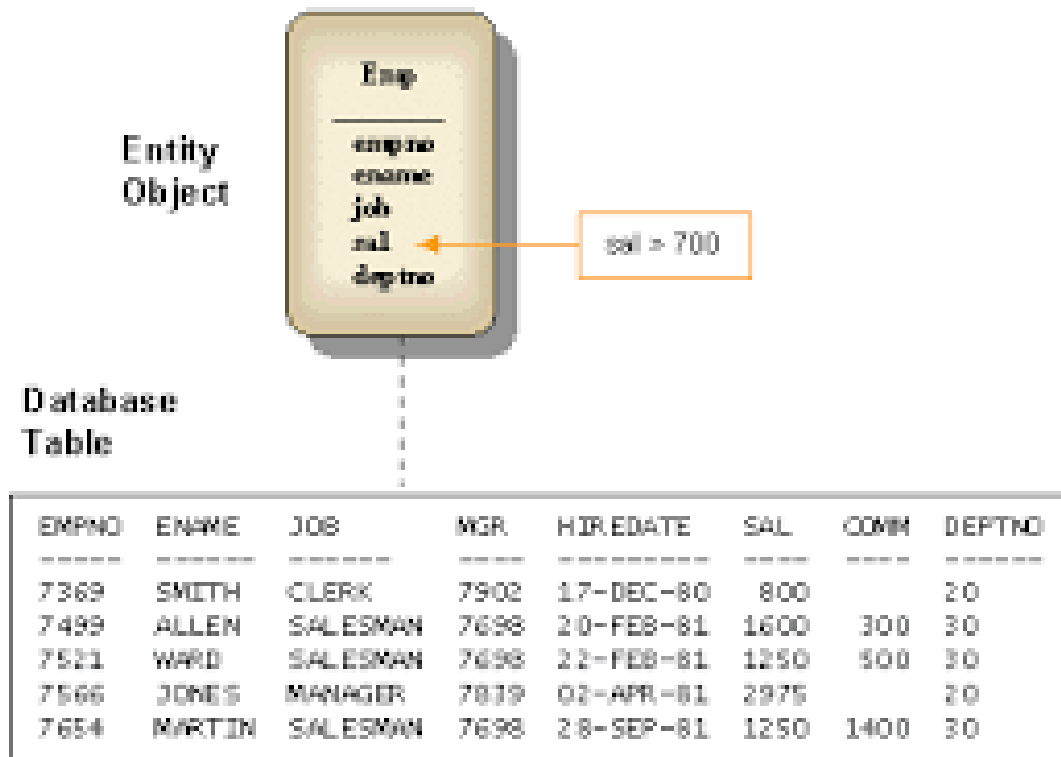
Step 1 - Entities

- Transforming an ER entity E:
 - Create a table for each entity.
 - The table name is E
 - The columns of this table are the attributes of the entity type.
 - The primary key of the table is the primary key of the entity type.
 - If E's key is composite, so will be the relational key.
 - If E has no key, add an artificial key to the table.

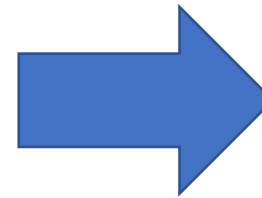
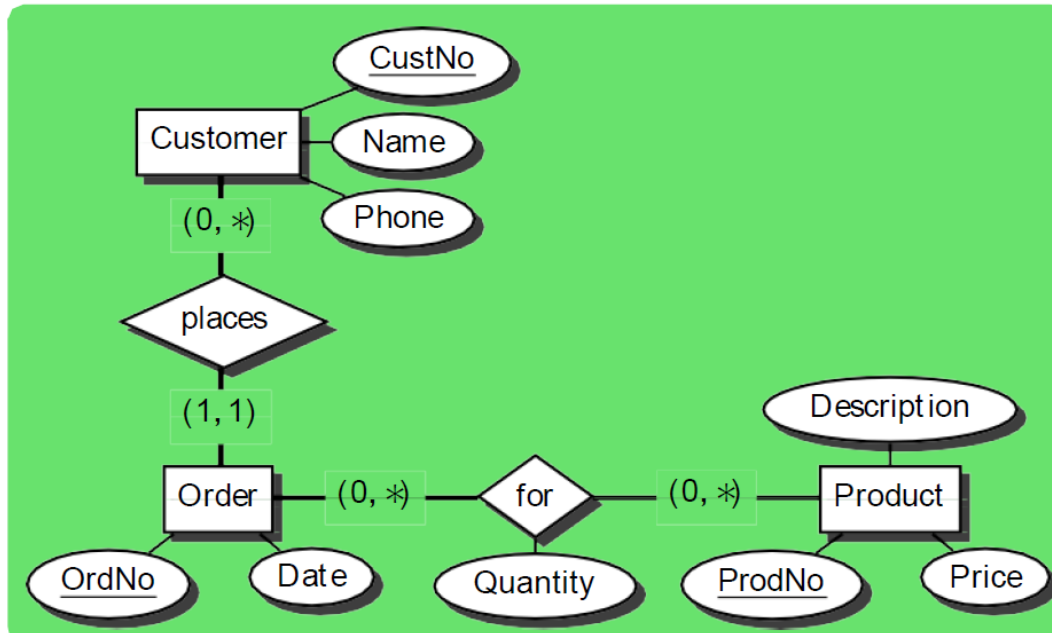


Step 1 - Entities

- Transforming an ER entity E:
 - Create a table for each entity.
 - The table name is E
 - The columns of this table are the attributes of the entity type.
 - The primary key of the table is the primary key of the entity type.
 - If E's key is composite, so will be the relational key.
 - If E has no key, add an artificial key to the table.



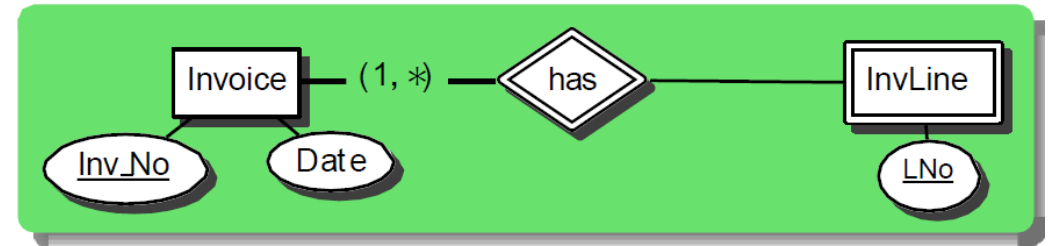
Example



Customers			Orders	
<u>CustNo</u>	Name	Phone	<u>OrdNo</u>	Date
10	Jones	624-9404	200	2/15/04
11	Smith		201	2/16/04

Products		
<u>ProdNo</u>	Description	Price
1	Apple	0.50
2	Kiwi	0.25
3	Orange	0.60

Weak entities



InvLine (LNo, Inv No → Invoice, ...)

When a weak entity is translated,
the key attributes of the owner
entity are added as a key and
foreign key

Step 2 – Transforming One to Many Relationships

- Transforming a relationship R:
 - Example: Customer–(0, *)–places–(1, 1)–Order
 - “one customer places many orders.”
 - In this case, add the key of the “one” side as a column to the “many” table to implement R.
 - This column will be a foreign key referencing a row in the table representing the related entity.
 - Convention: use relationship and role to name foreign key column

Orders (OrdNo, Date, CustNo → Customers)

Orders		
<u>OrdNo</u>	Date	CustNo
200	2/15/04	11
201	2/16/04	11

Customers		
<u>CustNo</u>	Name	Phone
10	Jones	624-9404
11	Smith	

Step 2 – Transforming One to Many Relationships

- Transforming a relationship R:
 - If the minimum cardinality is 1 on the many side, null values are not allowed in the foreign key column (column placed by in example).
 - If the minimum cardinality is 0, null values are allowed in the foreign key column.
 - The foreign key is null for those entities that do not participate in R at all.

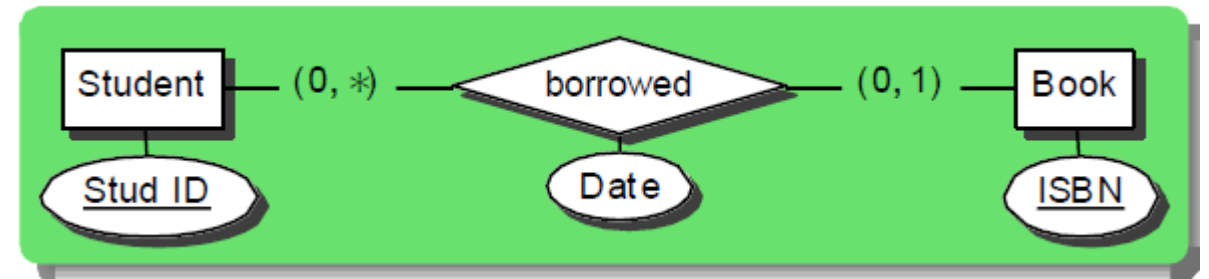
Orders (OrdNo, Date, CustNo → Customers)

Orders		
<u>OrdNo</u>	Date	CustNo
200	2/15/04	11
201	2/16/04	11

Customers		
<u>CustNo</u>	Name	Phone
10	Jones	624-9404
11	Smith	

Step 2 – Transforming One to Many Relationships

- Transforming one to many relationship with attributes:
 - Create an extra table that holds the key values of the related entities plus the relationship attributes.



`borrowed_by (ISBN → Books, Stud ID → Students, Date)`

Step 3 – Transforming Many to Many Relationships

- If R has maximum cardinality * on both sides, R is many-to-many.
 - Example: Order–(1, *)–for–(0, *)–Product, “an order contains many products, a product may be part of many orders.”
- R becomes its own table.
- The columns of this table are the keys of both participating entity types.
- These columns act as foreign keys referencing the entities and, at the same time, together form a composite key for the extra table.

for (OrdNo → Orders, ProdNo → Products, Quantity)

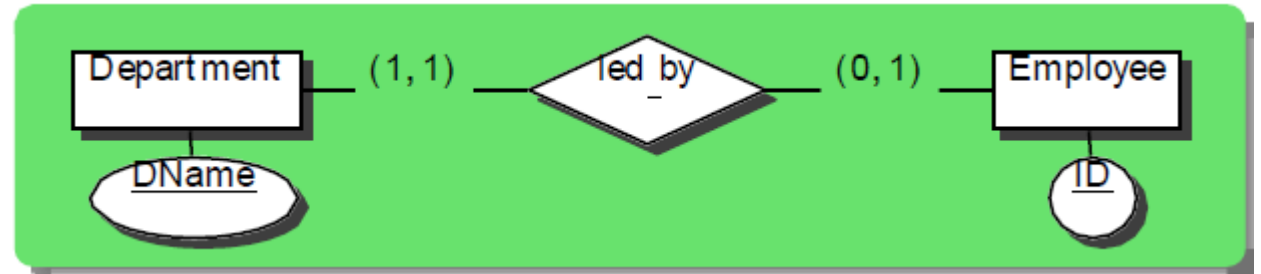
for		
<u>OrdNo</u>	<u>ProdNo</u>	Quantity
200	1	1
200	2	1
201	1	5

Orders		
<u>OrdNo</u>	Date	CustNo
200	2/15/04	11
201	2/16/04	11

Products		
<u>ProdNo</u>	Description	Price
1	Apple	0.50
2	Kiwi	0.25
3	Orange	0.60

Step 4 – Transforming One to One Relationships

- If R has maximum cardinality 1 on both sides, R is one-to-one.
- We can essentially transform as if R were one-to-many, but additional key constraints are generated.
- E.g.
- Department is led by an employee
 - *“Every department is led by exactly one employee”*
- We may declare the foreign key led by as NOTNULL. (This is not possible if led by is hosted in the Employee table.)



Department (DName, ..., led by → Employee)

Advanced Databases

Normalization



Normalization is a design technique	Objectives	Focus	How	Why is it important?
<ul style="list-style-type: none"> “..a very important ingredient in database design”, Coronel, C., & Morris, S. (2016). <i>Database systems: design, implementation, & management</i>. Cengage Learning. 	<ul style="list-style-type: none"> Eliminate redundant data (storing the same data in more than one table) Ensure data dependencies make sense (only storing related data in a table) 	<ul style="list-style-type: none"> Correct assignment of attributes to tables. 	<ul style="list-style-type: none"> Considering the rules of the real world Examining the actual values attributes can take 	<ul style="list-style-type: none"> If you don't normalize, databases can be inaccurate, slow, and inefficient for day to day transactions and they might not produce the data you expect.

Database Normalization

Anomalies

- Update anomaly
 - We have two rows for employee Rick as he works in two departments of the company.
 - If we want to update Rick's address then we have to update two rows or the data will become inconsistent.
 - If somehow, the correct address gets updated for one department but not the other then as far as the database is concerned Rick has two different addresses – this is not correct or consistent with the real world we are modelling.

emp_id	emp_name	emp_address	emp_dept
101	Rick	Dublin 6	D001
101	Rick	Dublin 6	D002
123	Maggie	Dublin 7	D890
166	Glenn	Dublin 8	D900
166	Glenn	Dublin 8	D004

Anomalies

- Insert anomaly
 - Suppose a new employee joins the company
 - They are not assigned to any department while they are training and there is no official training department
 - At the moment we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

emp_id	emp_name	emp_address	emp_dept
101	Rick	Dublin 6	D001
101	Rick	Dublin 6	D002
123	Maggie	Dublin 7	D890
166	Glenn	Dublin 8	D900
166	Glenn	Dublin 8	D004

Anomalies

- Delete anomaly
 - Suppose the company closes the department D890
 - Deleting the rows that have emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

emp_id	emp_name	emp_address	emp_dept
101	Rick	Dublin 6	D001
101	Rick	Dublin 6	D002
123	Maggie	Dublin 7	D890
166	Glenn	Dublin 8	D900
166	Glenn	Dublin 8	D004

Exercise (Dental Appointments)

staffNo	dentistName	patientNo	patientName	appointment date time	surgeryNo
S1011	Tony Smith	P100	Gillian White	12-Aug-03 10.00	S10
S1011	Tony Smith	P105	Jill Bell	13-Aug-03 12.00	S15
S1024	Helen Pearson	P108	Ian MacKay	12-Sept-03 10.00	S10
S1024	Helen Pearson	P108	Ian MacKay	14-Sept-03 10.00	S10
S1032	Robin Plevin	P105	Jill Bell	14-Oct-03 16.30	S15
S1032	Robin Plevin	P110	John Walker	15-Oct-03 18.00	S13

- Provide examples of insert, update and delete anomalies

Database Tables and Normalization

- Normalization works through a series of stages called normal forms:
 - 1NF (First Normal Form)
 - 2NF (Second Normal Form)
 - **3NF (Third Normal Form)**
 - BCNF (Boyce-Codd Normal Form)
 - 4NF (Fourth Normal Form)
 - 5NF (Fifth Normal Form)
 - 6NF (Sixth Normal Form)
- The higher levels of normalization are not always advisable.



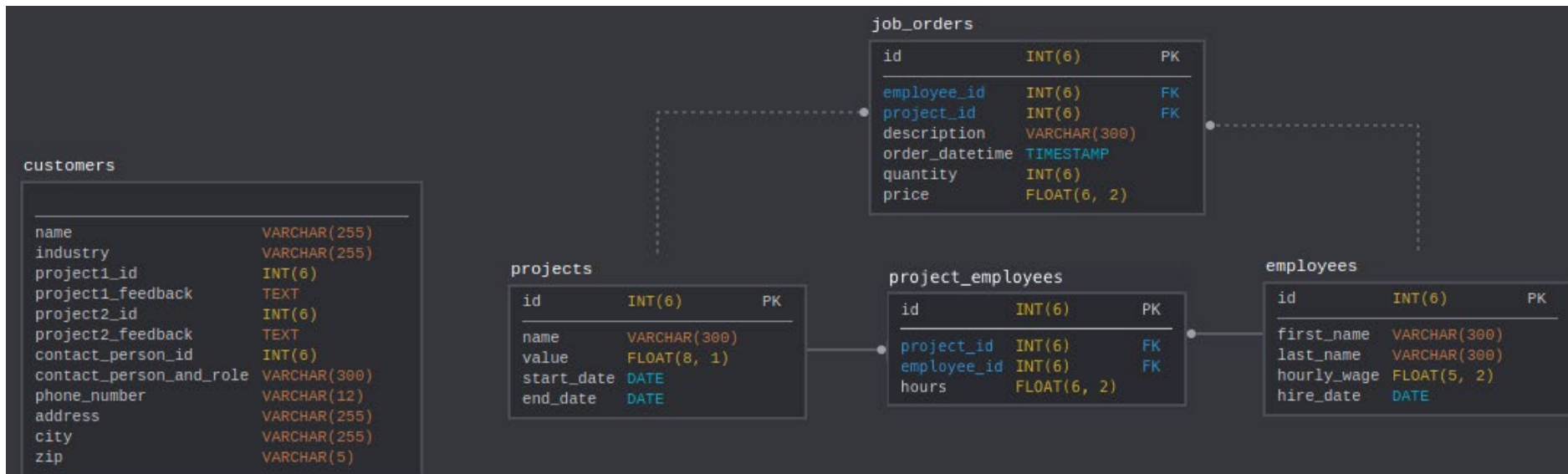
What is good about normalization?

- Objectives of DB Design
 - Arrange data into logical groups such that each group describes a small part of the world represented by the data
 - Minimize the amount of duplicated data stored in a database
 - Build a database which allows you to access and manipulate the data quickly and efficiently balanced with maintaining the integrity of the data stored
 - Organise the data so that, when you modify it, you make the changes in only one place



What is good about normalization?

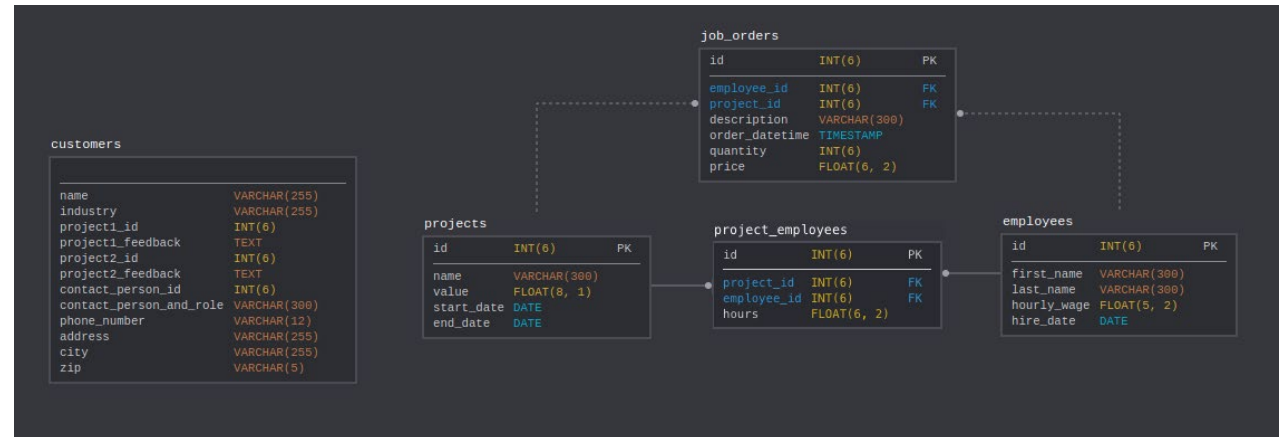
- Updates run quickly due to no data being duplicated in multiple locations.
- Inserts run quickly since there is only a single insertion point for a piece of data and no duplication is required.
- Tables are typically smaller than the tables found in non-normalized databases.
 - This usually allows the tables to fit into the buffer, thus offering faster performance.
- Data integrity and consistency is an absolute must if the database must be ACID compliant.
 - A normalized database helps a lot with this.



- Codey's Construction Company
- Database contains the tables: projects, job_orders, employees, and project_employees
- Customers table has just been added

First Normal Form (1 NF)

- Objective:
 - Data is stored in tables with rows uniquely identified by a primary key
 - Data within each table is stored in individual columns in its most reduced form
 - There are no repeating groups



First Normal Form (1 NF)

- Codey's Construction's table customers violates all three rules of 1NF.
 - There is no primary key
 - Need to look up by name which is not guaranteed to be unique

customers

name	VARCHAR(255)
industry	VARCHAR(255)
project1_id	INT(6)
project1_feedback	TEXT
project2_id	INT(6)
project2_feedback	TEXT
contact_person_id	INT(6)
contact_person_and_role	VARCHAR(300)
phone_number	VARCHAR(12)
address	VARCHAR(255)
city	VARCHAR(255)
zip	VARCHAR(5)

First Normal Form (1 NF)

- Codey's Construction's table customers violates all three rules of 1NF.
 - The data is not in its most reduced form.
 - The column contact_person_and_role can be further divided into two columns, such as contact_person and contact_role.

customers

name	VARCHAR(255)
industry	VARCHAR(255)
project1_id	INT(6)
project1_feedback	TEXT
project2_id	INT(6)
project2_feedback	TEXT
contact_person_id	INT(6)
contact_person_and_role	VARCHAR(300)
phone_number	VARCHAR(12)
address	VARCHAR(255)
city	VARCHAR(255)
zip	VARCHAR(5)

First Normal Form (1 NF)

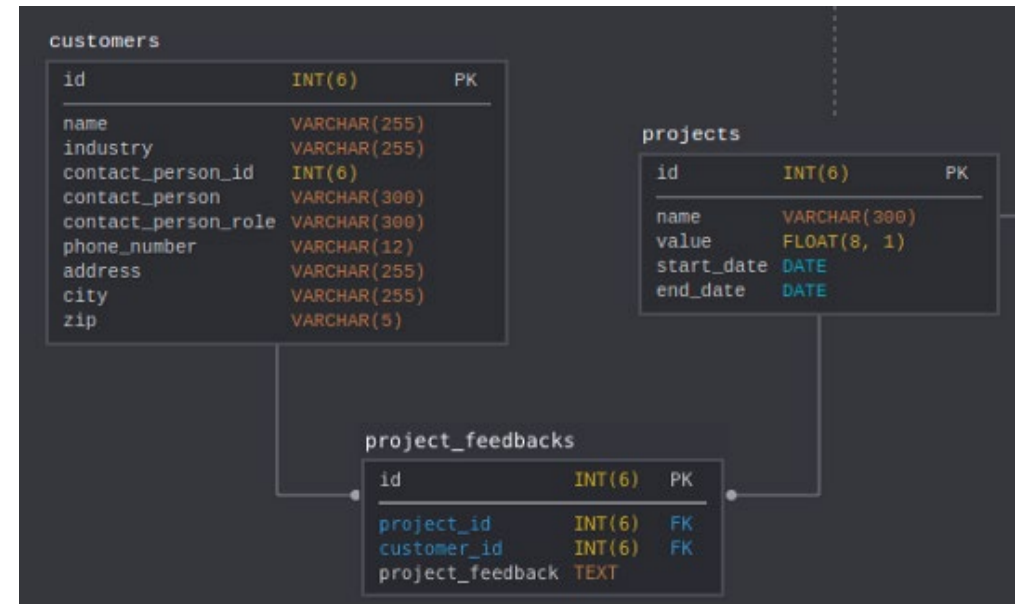
- Codey's Construction's table customers violates all three rules of 1NF.
 - There are two repeating groups of columns
 - (project1_id, project1_feedback) and (project2_id, project2_feedback).

customers

name	VARCHAR(255)
industry	VARCHAR(255)
project1_id	INT(6)
project1_feedback	TEXT
project2_id	INT(6)
project2_feedback	TEXT
contact_person_id	INT(6)
contact_person_and_role	VARCHAR(300)
phone_number	VARCHAR(12)
address	VARCHAR(255)
city	VARCHAR(255)
zip	VARCHAR(5)

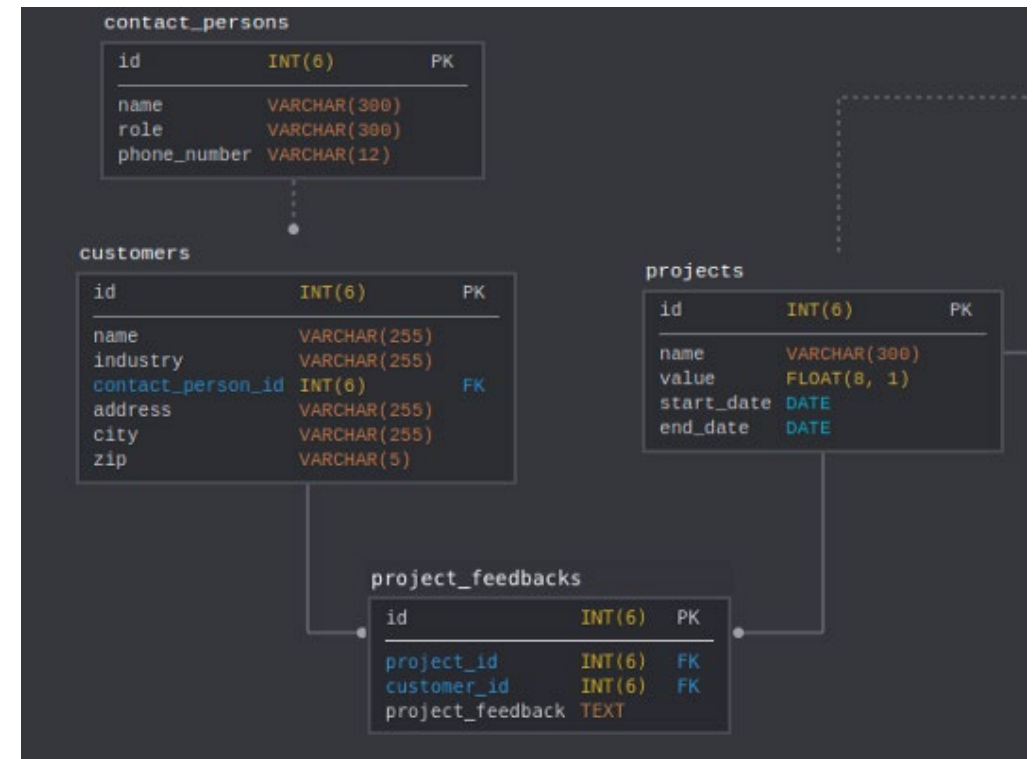
First Normal Form (1 NF)

- To bring to 1NF:
 - We need to add a primary key column called ID
 - We will split contact person and role into two separate columns : contact_person, contact_person_role
 - Move the repeating groups (project id and feedback) to a new table call project_feedbacks (taking account of the project table that already exists)



Second Normal Form (2 NF)

- All data (non-prime attributes) in each table must relate directly to the record that the primary key of the table (prime attribute)
 - contact_person, contact_person_role and phone number in this organisation are dependent on the contact_person_id
 - So we should extract these into a new table contact_persons (and give this a primary key to fulfil 1NF)



Third Normal Form (3 NF)

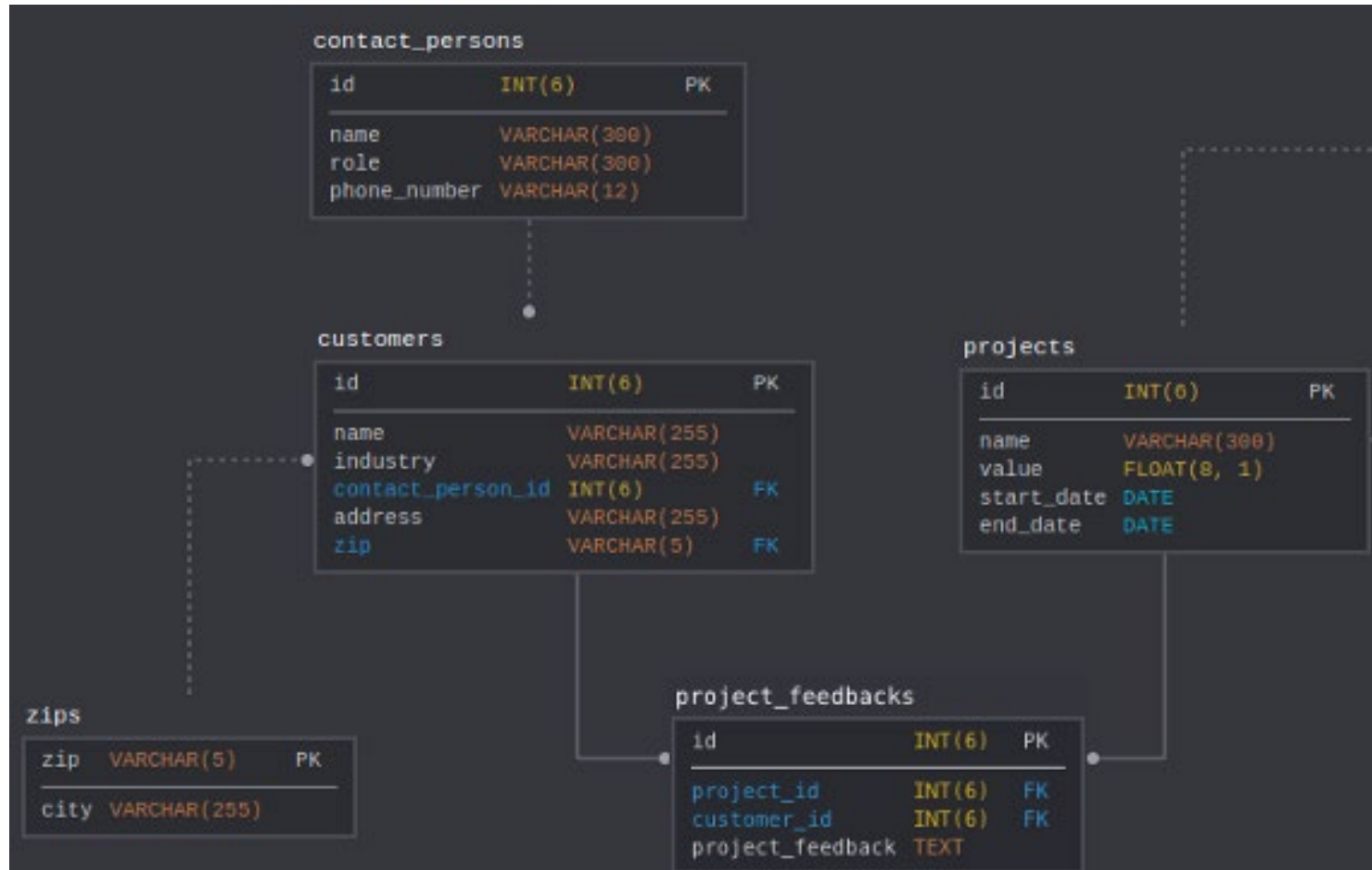
id	name	industry	contact_id	address	city	zip
000001	Next University	Education	000001	1 Coding Lane	Next	99999
000002	XYZ Health Center	Healthcare	000002	88 Hospital Avenue	Healersville	55555
000003	ArchiTECHS	Architecture	000003	77 Tower Street	Gridlock	12345

- This is how our customers table looks after 1NF and 2NF:
- To be in 3NF we need to remove **transitive dependencies**:
 - When one non-prime attribute is dependent on another non-prime attribute

Third Normal Form (3 NF)

id	name	industry	contact_id	address	city	zip
000001	Next University	Education	000001	1 Coding Lane	Next	99999
000002	XYZ Health Center	Healthcare	000002	88 Hospital Avenue	Healersville	55555
000003	ArchiTECHS	Architecture	000003	77 Tower Street	Gridlock	12345

- Look at city and zip:
 - City relies on the customer but also on the zip
- We could, potentially, if a customer moves city update one column but not the other.



Third Normal Form (3 NF)

To overcome this we create a new table (zips) to handle the dependency between city and zip

Functional Dependencies

- If you say there is a dependency between attributes in a relation this is the same as saying that there is a functional dependency between those attributes.
 - For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y.
 - E.g. If there is a dependency in a database such that attribute B is dependent upon attribute A, you would write this as:
 - $A \rightarrow B$
 - E.g. Student ID \rightarrow Student Name (a student's name can be uniquely determined from their ID)
-

Transitive Dependencies

- Transitive dependencies occur when there is an indirect relationship that causes a functional dependency.
 - A transitive dependency exists when you have the following functional dependency pattern:
 - $A \rightarrow B, B \rightarrow C$ so $A \rightarrow C$
 - $A \rightarrow C$ is a transitive dependency when it is true only because both $A \rightarrow B$ and $B \rightarrow C$ are true.
 - Suppose we have a non-normalised table for a warehouse:
 - For each item in the warehouse we store the item number, number assigned to the distributor of that item (distrib number) and the phone number of that distributor (distrib_phone_number)
 - The only reason that the distributor phone number is functionally dependent on the item number is because the distributor is functionally dependent on the item number and the phone number is functionally dependent on the distributor. The functional dependencies are really:
 - $\text{Item_numb} \rightarrow \text{distrib_numb}$
 - $\text{Distrib_numb} \rightarrow \text{distrib_phone_number}$
-



Domain Key Normal Forms

- Boyce-Codd normal form (BCNF), fourth normal form (4NF), and fifth normal form (5NF) are examples of Domain Key Normal Forms.
 - Each form eliminates a possible modification anomaly but doesn't guarantee prevention of all possible modification anomalies.
-

Boyce-Codd Normal Form (BCNF)

When a table has more than one candidate key, anomalies may result even though the relation is in 3NF.

Boyce-Codd normal form is a special case of 3NF (3.5NF)

A table is in BCNF if, and only if:

1. It is in the **Third Normal Form**.
2. And, for any functional dependency $A \rightarrow B$, A should be a **super key** (set of attributes that uniquely identifies each tuple of a relation)

Boyce-Codd Normal Form (BCNF)

- Each Student may major in several subjects.
- For each Major, a given Student has only one Advisor.
- Each Major has several Advisors.
- Each Advisor advises only one Major.
- Each Advisor advises several Students in one Major.

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

Boyce-Codd Normal Form (BCNF)

- The functional dependencies for this table are listed below. The first one is a candidate key; the second is not.
- $\text{Student_id, Major} \twoheadrightarrow \text{Advisor}$
- $\text{Advisor} \twoheadrightarrow \text{Major}$

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

Boyce-Codd Normal Form (BCNF)

- Anomalies for this table include:
 - Delete – student deletes advisor info
 - Insert – a new advisor needs a student
 - Update – inconsistencies
- **Note:** No single attribute is a candidate key.
- PK can be Student id, Major or Student id, Advisor.

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

Boyce-Codd Normal Form (BCNF)

1. **St_Adv** (Student id, Advisor)

2. **Adv_Maj** (Advisor, Major)

- This is now in BCNF

Normal Forms and Dependencies

- For a table to be in second normal form (2NF), there must be no case of a nonprime attribute in the table that is functionally dependent upon a subset of a candidate key.
- For a table to be in third normal form (3NF), every nonprime attribute must have a nontransitive functional dependency on every candidate key.
- For a table to be in Boyce-Codd Normal Form (BCNF), every functional dependency (other than trivial dependencies) must be on a **superkey**.
 - Superkey = a set of attributes that uniquely identifies each tuple of a relation

Exercises

- **Student_Grade_Report** (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)
- A student can take many courses
- An instructor can deliver many courses
- Each course is delivered by one instructor
- Bring to 3NF

Normalisation: Advantages



Increases data consistency as it avoids the duplication of data by storing the data in one place only.



Helps in grouping like or related data under the same schema, thereby resulting in the better grouping of data.



Improves searching faster as indexes can be created faster.
(Very useful for OLTP (online transaction processing)).

Normalisation: Disadvantages

Storing data in one place causes a delay when retrieving

e.g. We cannot find the associated data for, say a product or employee in one place, so we need to join more than one table.



Means that normalization is not a good option in OLAP transactions (online analytical processing).