SQL DATABASE DESIGN

CMPE2400

Credits to:

- JD Silver
- Steven Dytiuk

DATABASE DESIGN

- When designing a new database, you must consider…
  - The tables that belong to the database
  - The columns that belong in each table
  - How tables and columns interact with each other
- Database design is concerned with the logical structure

DATABASE DESIGN

01

You must determine your needs by examining the data to be stored

02

Specify the entities to be stored

03

Determine the relationships between the entities

04

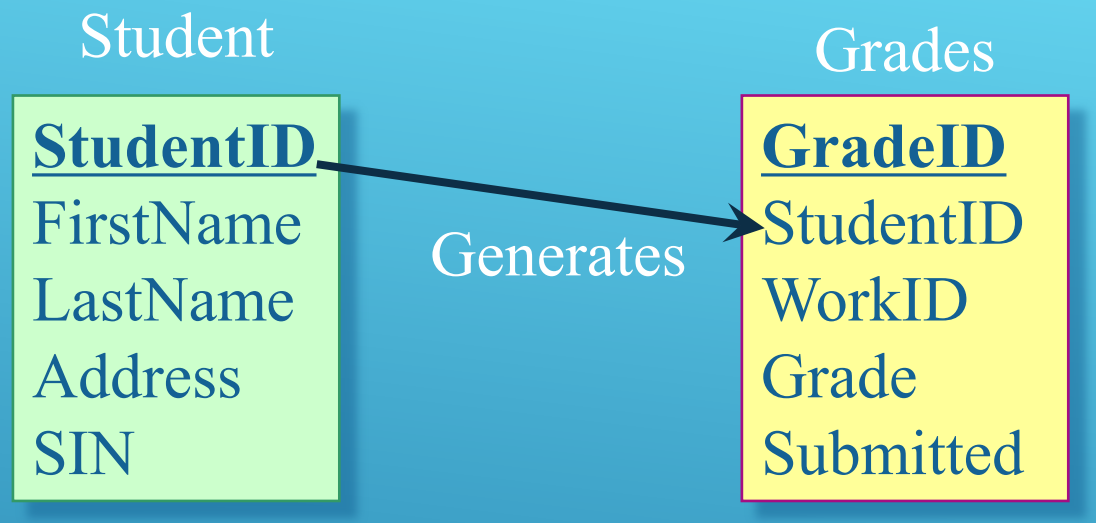
Normalize the tables to improve efficiency using no loss decomposition

BASIC STEPS

- Examine the current business process
- For example, student marks.
  - Grades are entered per student for each assignment
  - Each assignment has a grade value in the overall mark, and a due date
  - Desire final marks for students, and summary per assignment

MAKE A LIST OF ENTITIES

- Determine what you should store for each entity as attributes
  - Student
    Name, ID number
  - Work
    Name, Category, Out of, Due date
  - Grade
    Student Name, Work, Mark, Date Submitted
  - Attendance
    Date, Present, Late



- An entity (here a student) has a relationship with another entity, in this case a grade

E-R DIAGRAMS

PRIMARY KEYS

- Determine a unique value that can be used to identify each entity
- Sometimes a *natural* primary key exists, such as a SIN or Student ID number
- You may need to generate unique values to use as a primary key when none exists (*surrogate*)

PRIMARY KEYS FOR STUDENT GRADES

The following primary keys were used for a DB to store student marks

Student	natural - StudentID number
Grade	surrogate - autoincrement number
Work	surrogate - autoincrement number

When a table is drawn, the primary key values are shown as underlined

RELATIONSHIPS

- Relationships are formed between entities by storing the key values in each table
- There can be several types of relationships
  - ***One to one***: each student is in one class (1:1)
  - ***One to many***: each student has several marks (1:N)
  - ***Many to many***: team projects might have several students working together on several assignments (M:N)

RELATIONSHIPS - CARDINALITY

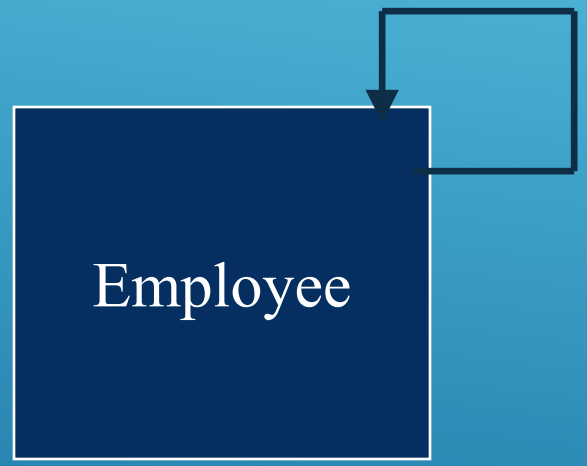
The *cardinality ratio* specifies the number of relationships that an entity can be involved in

The types are:

1 : 1 one to one	1 : N one to many	M : N many to many

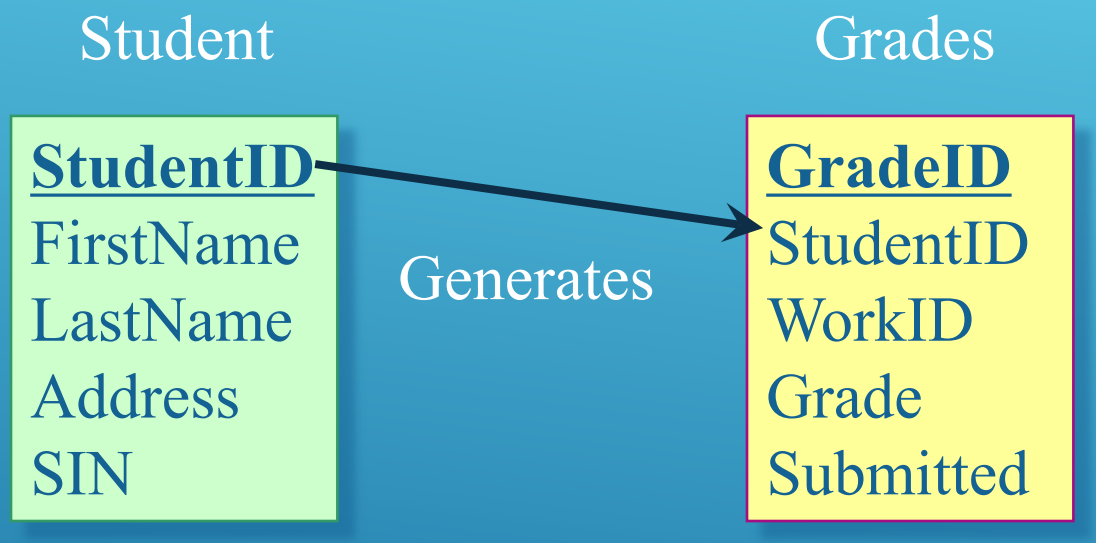
UNARY RELATIONSHIPS

- A *unary* relationship specifies the relationship between two of the same entities



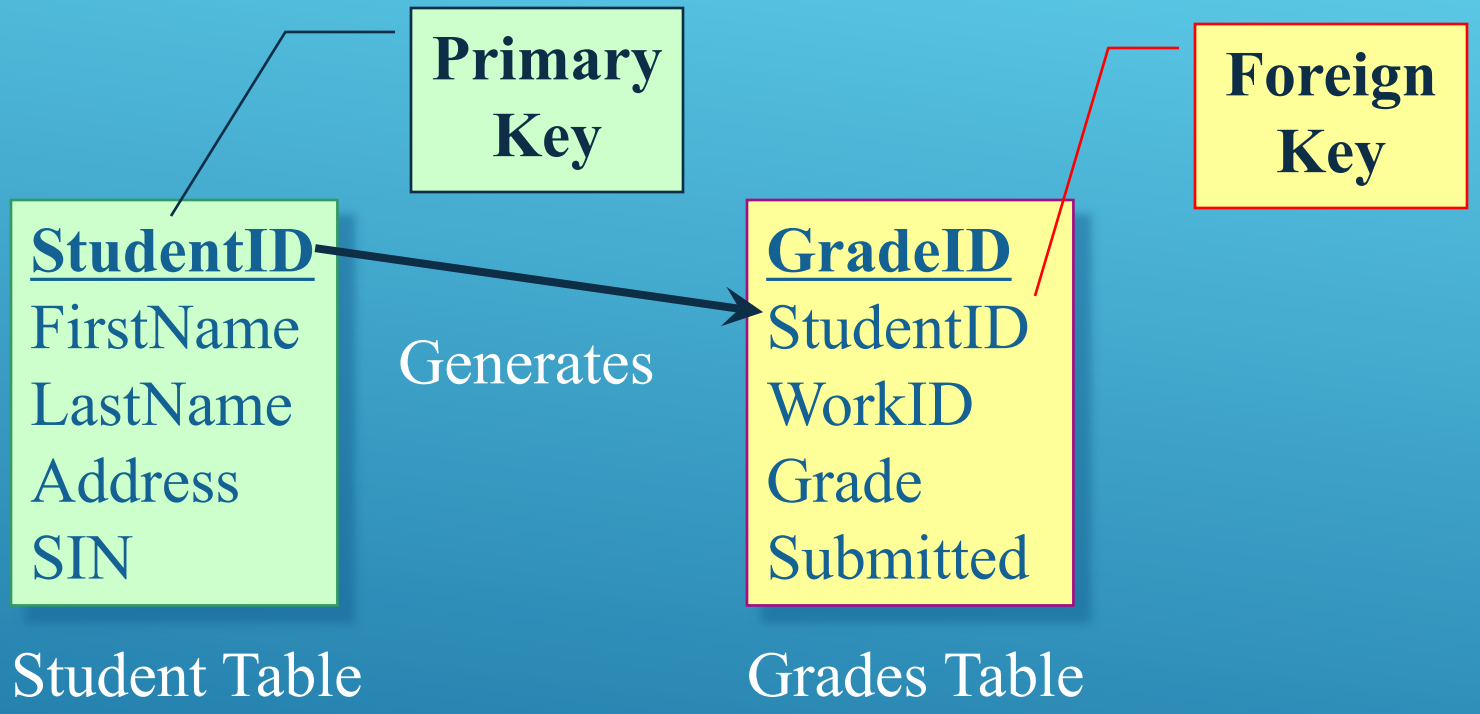
BINARY RELATIONSHIPS

- A *binary* relationship is one between two different entities



RELATIONSHIPS – USING KEYS

- The key values in each table are used to join the entities



RELATIONSHIPS – USING KEYS

THE PRIMARY KEY IS USED TO UNIQUELY IDENTIFY AN ENTITY IN A TABLE

ITS *VALUE* IS USED AS A FOREIGN KEY IN ANOTHER TABLE TO CONNECT THE ENTITIES IN RELATED TABLES

FOR EXAMPLE, THE STUDENTID IS USED IN THE GRADES TABLE TO CONNECT A MARK ENTRY TO A STUDENT

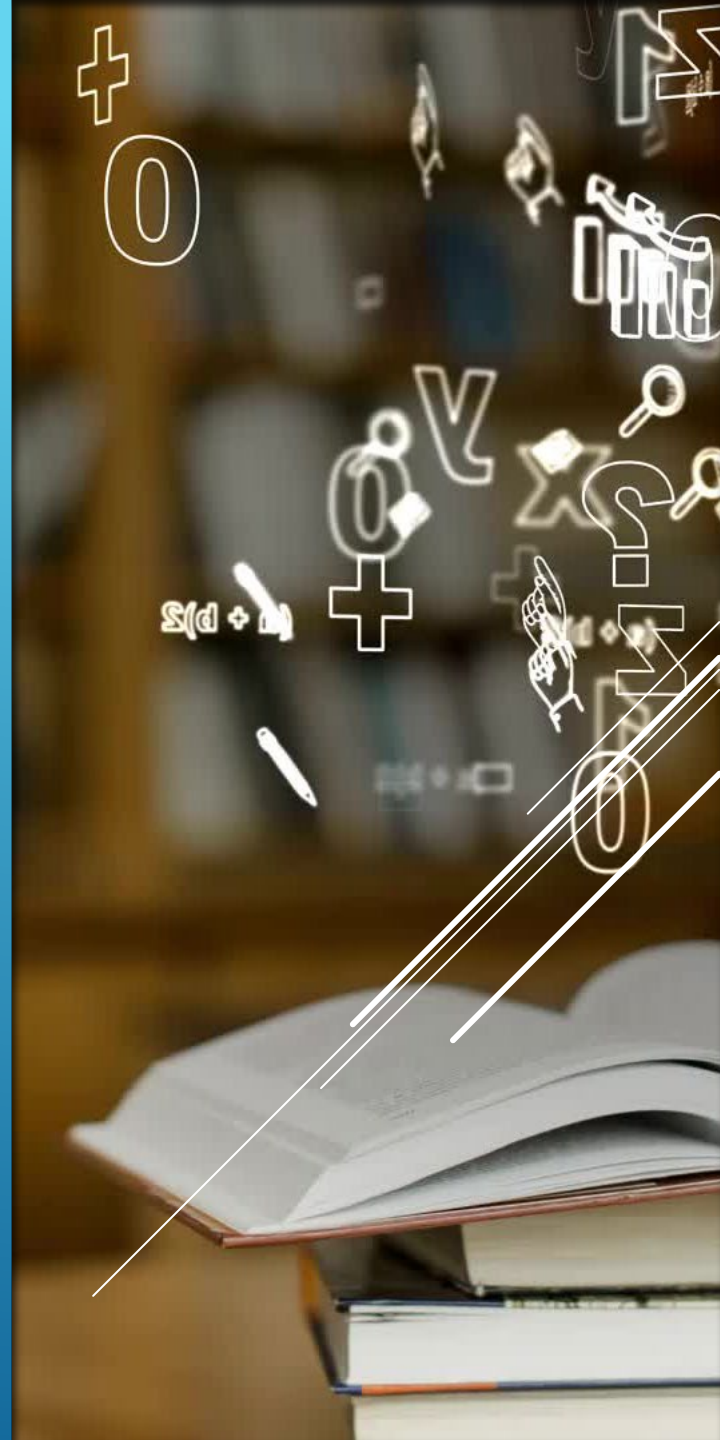
IN THE STUDENT TABLE, STUDENTID IS A *PRIMARY KEY*, BUT IT IS USED AS A *FOREIGN KEY* IN THE GRADES TABLE

- A primary key may be the combination of several data columns to form a composite key
- Data integrity would be guaranteed by ensuring that no duplicates of the key may exist
- Example: only one mark allowed per course per student; composite key would be student ID + course ID

COMPOSITE KEYS

E – R EXAMPLE

- We will develop an example database for a book publishing industry
- Important entities are…
  - Authors
  - Titles
  - Publishers
  - Editors



DETERMINE THE ATTRIBUTES

- Draw an E-R diagram with attributes
- Some attributes are not stored, but are *derived* (calculated) when needed

DETERMINE THE ATTRIBUTES

- Break down the attributes to the *atomic* level
- Some names can be split into first/last

Authors	***Titles***	***Publishers***	***Editors***
Au_fname	Name	Name	Ed_fname
Au_lname	Price	Address	Ed_lname
Address	PubDate		Address
Phone			Phone

DETERMINE KEY VALUES

- The primary key of each table must be unique, and is shown underlined in a table
- Names are not good key values

Authors
Au_id
Au_fname
Au_lname
Address
Phone

Titles
Title_id
Name
Price
PubDate

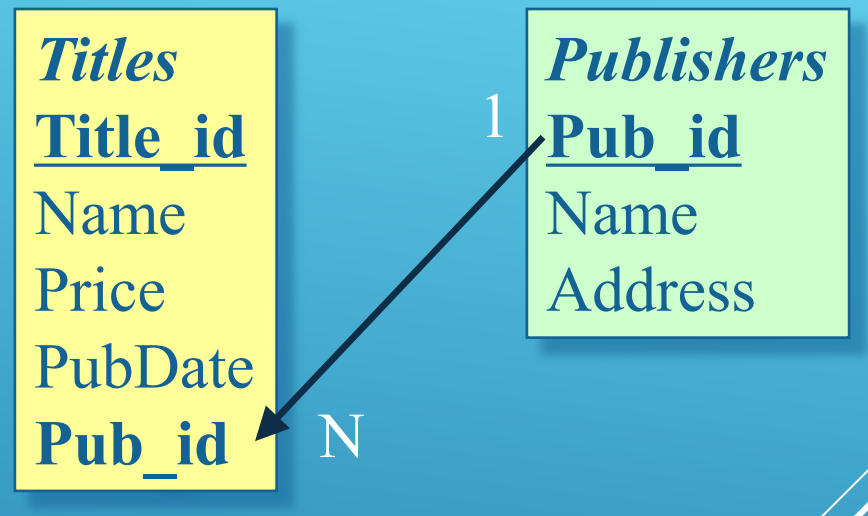
Publishers
Pub_id
Name
Address

Editors
Ed_id
Ed_fname
Ed_lname
Address
Phone

DETERMINE THE RELATIONSHIPS

Use a matching *Foreign Key* to connect the entities

For example, each title has a publisher, one publisher can have many titles



REFERENTIAL INTEGRITY

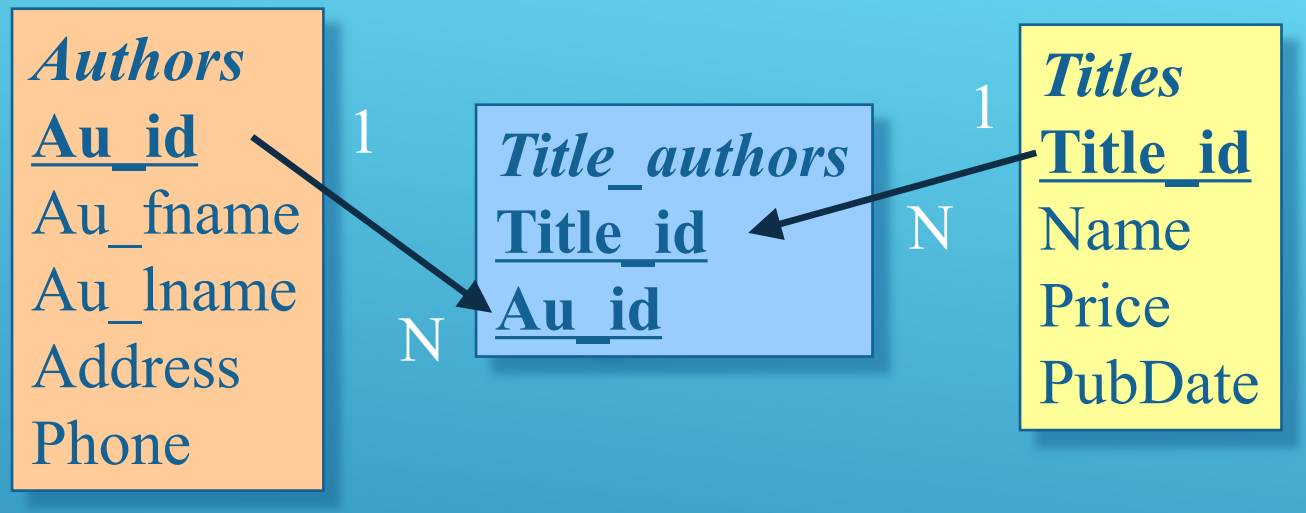
A foreign key must always have a matching primary key

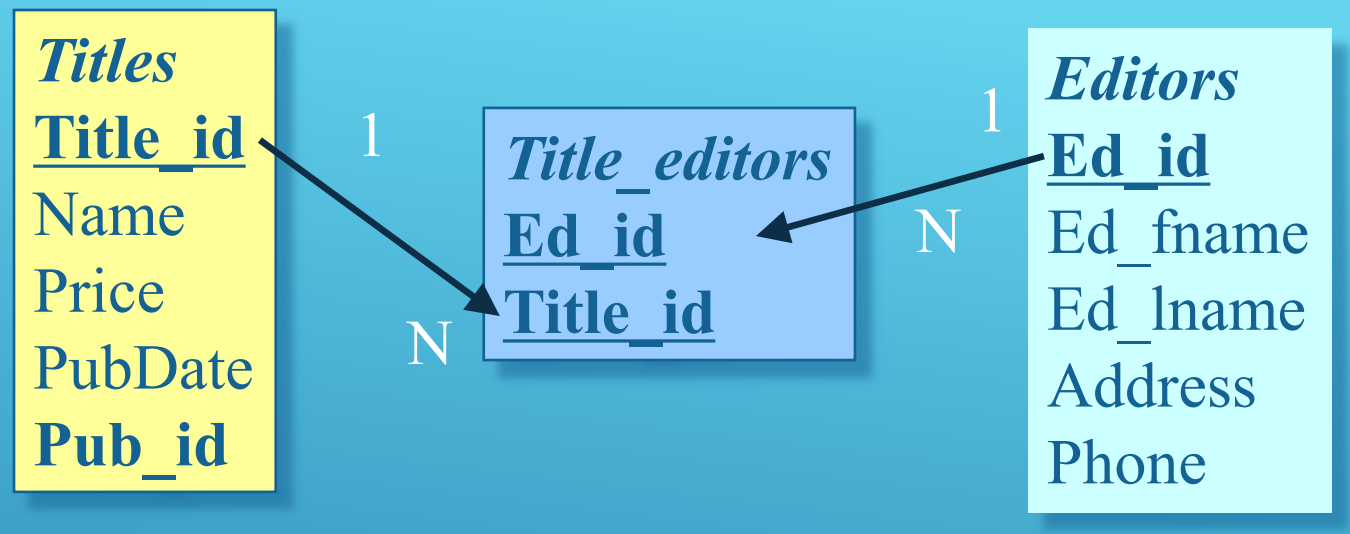
If a publisher's id changes, then all foreign key values in the titles table should be updated

If a new title is added to the titles table, then the system should verify that the pub_id exists

A MANY TO MANY RELATIONSHIP

- An author can write many books.
- A book can have many authors
- Primary key is Title_id + Au_id





An editor can edit many books, and a book can have multiple editors

The primary key is a composite key composed of Ed_id and Title_id

ANOTHER N:M RELATIONSHIP

E-R GUIDELINES

Represent each entity as a table

Represent each property of an entity as a column in a table

Create a primary key (PK) for each table

Locate 1:N relationships and ensure that there is a PK for every foreign key (FK)

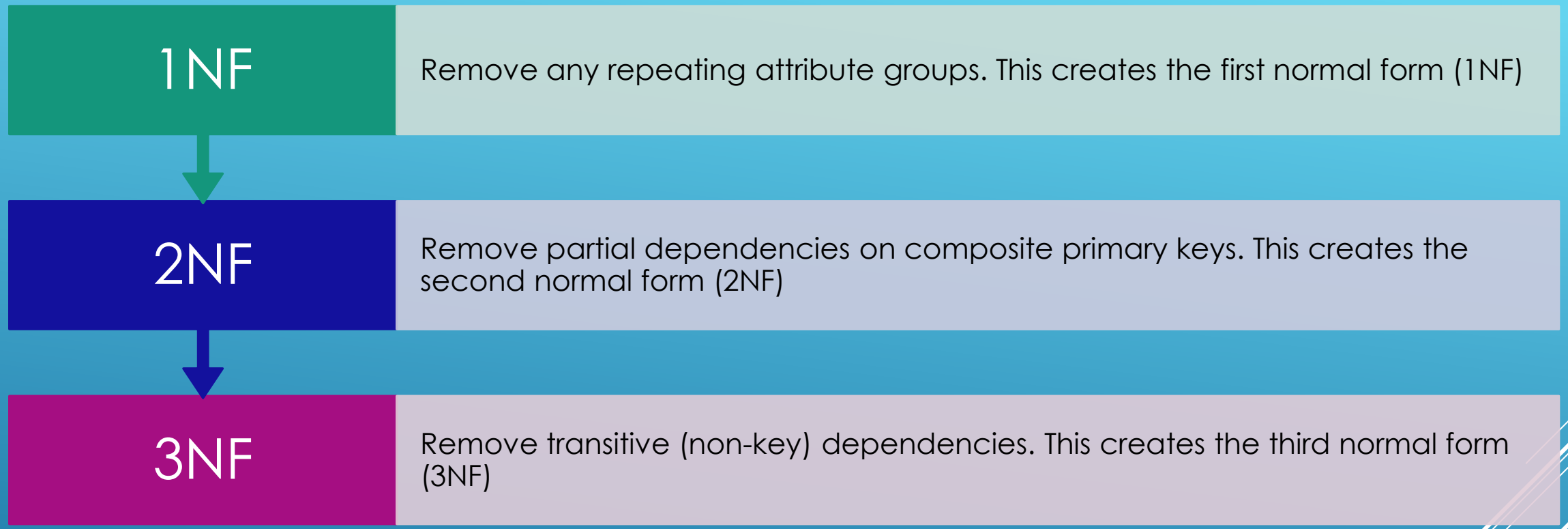
Represent N:M relationships with a connecting table

Collapse 1:1 relationships into one table

NEXT STEP - NORMALIZATION

- The next step of analysis uses what is termed *normal forms*
- A database is reformatted into a progression of normal forms which represent different degrees of optimization
- The normal forms are guidelines for relational database optimization

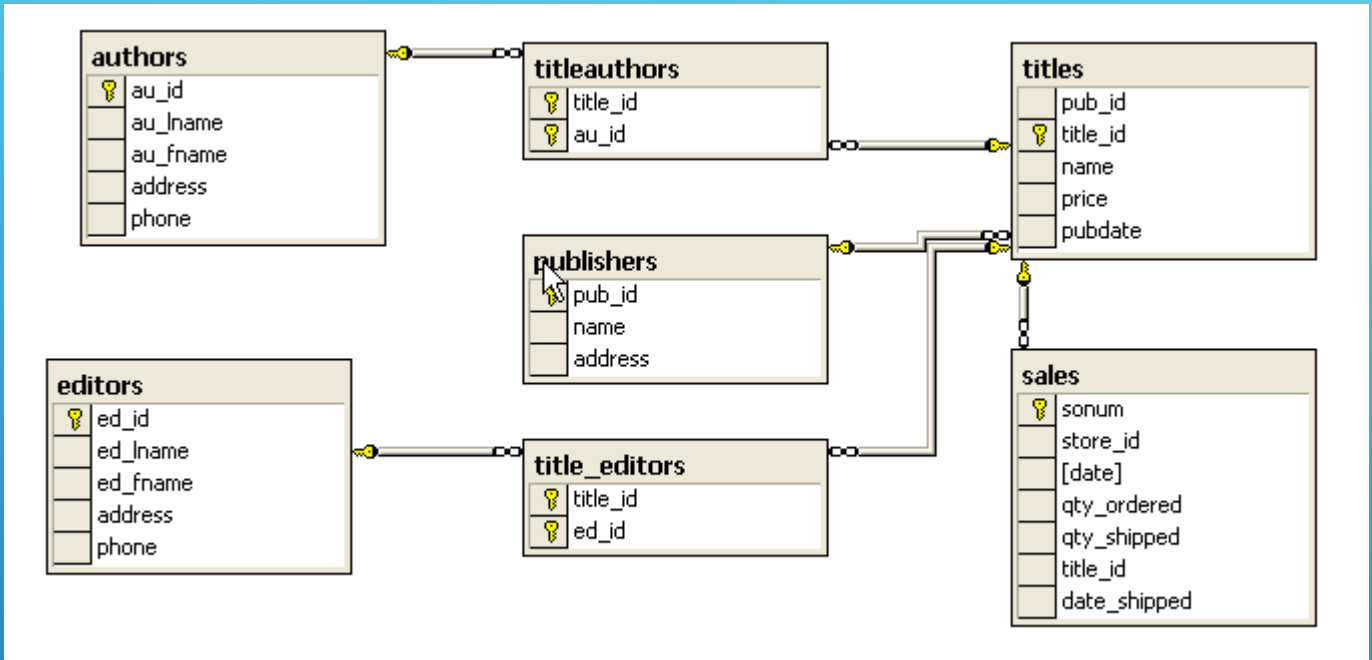
NORMALIZATION STEPS



ZERO NORMAL FORM (0NF)

The zero normal form is the result of our E-R analysis

There is no optimization in this form



ZERO NORMAL FORM

FIRST NORMAL FORM (1NF)

1NF requires that each value in the table be *atomic*

- For example, name should be stored as first name and last name

1NF requires that there should not be *repeating attribute groups*

FIRST NORMAL FORM (1NF)

1NF requires all data to be *atomic*

In our database, the address column should be separated into address, city, province and postal code fields

A table cannot contain repeating attribute groups

1NF – REPEATING ATTRIBUTE GROUPS

- While the atomicity of items can be seen easily from the given entities, the repeating attribute groups can only be seen if we try to see how actual data will be stored.
- Let's consider the **sales** table. For each sales order, we have a unique sales order number (sonum). A sales order is for one store, for a given date, but it will typically have multiple orders (title_id, qty_ordered, qty_shipped, date_shipped).
- This can be shown as in the next slide.

Sonum	Store_id	Date	Title_id-1	qty_ordered-1	qty_shipped-1	Date_shipped-1	Title_id-2	Qty_ordered-2	Qty_shipped-2	Date_shipped-2
12342	ED21	2021-02-15	BU1032	25	20	2021-03-01	RV4135	50	50	2021-03-03
12453	ED15	2021-02-20	TR2176	100	80	2021-03-01	BU1032	30		
12654	ED30	2021-03-01	RV4135	80	80	2021-03-05	TR2176	40	35	2021-03-03

1NF – REPEATING ATTRIBUTE GROUPS

Sonum	Store_id	Date	Title_id	Qty_ordered	Qty_shipped	Date_shipped
12342	ED21	2021-02-15	BU1032	25	20	2021-03-01
12342	ED21	2021-02-15	RV4135	50	50	2021-03-03
12453	ED15	2021-02-20	TR2176	100	80	2021-03-01
12453	ED15	2021-02-20	BU1032	30		
12654	ED30	2021-03-01	RV4135	80	80	2021-03-05
12654	ED30	2021-03-01	TR2176	40	35	2021-03-03

SECOND NORMAL FORM (2NF)

THE SECOND NORMAL FORM REQUIRES THAT EVERY NON-KEY COLUMN BE DEPENDANT ON THE *ENTIRE* PRIMARY KEY

THIS MEANS THAT YOU MUST CAREFULLY EXAMINE SITUATIONS WHERE YOU USE A COMPOSITE PRIMARY KEY

2NF ONLY APPLIES WHEN A COMPOSITE KEY IS INVOLVED

2 NF –PARTIAL DEPENDENCIES

In the previous sales table, the quantity ordered, quantity shipped, date shipped in a row are for a given Sonum and the title_id. Thus these 2 together form the primary key of the table. We have a composite primary key.

What about the stor_id and date. This combination is the same for a given Sonum, irrespective of the title_id.

So store_id and date are only dependent on Sonum. This is partial dependency

2NF – ELIMINATING PARTIAL DEPENDENCIES

1. Remove the partially dependent column from the original table and place into a new table

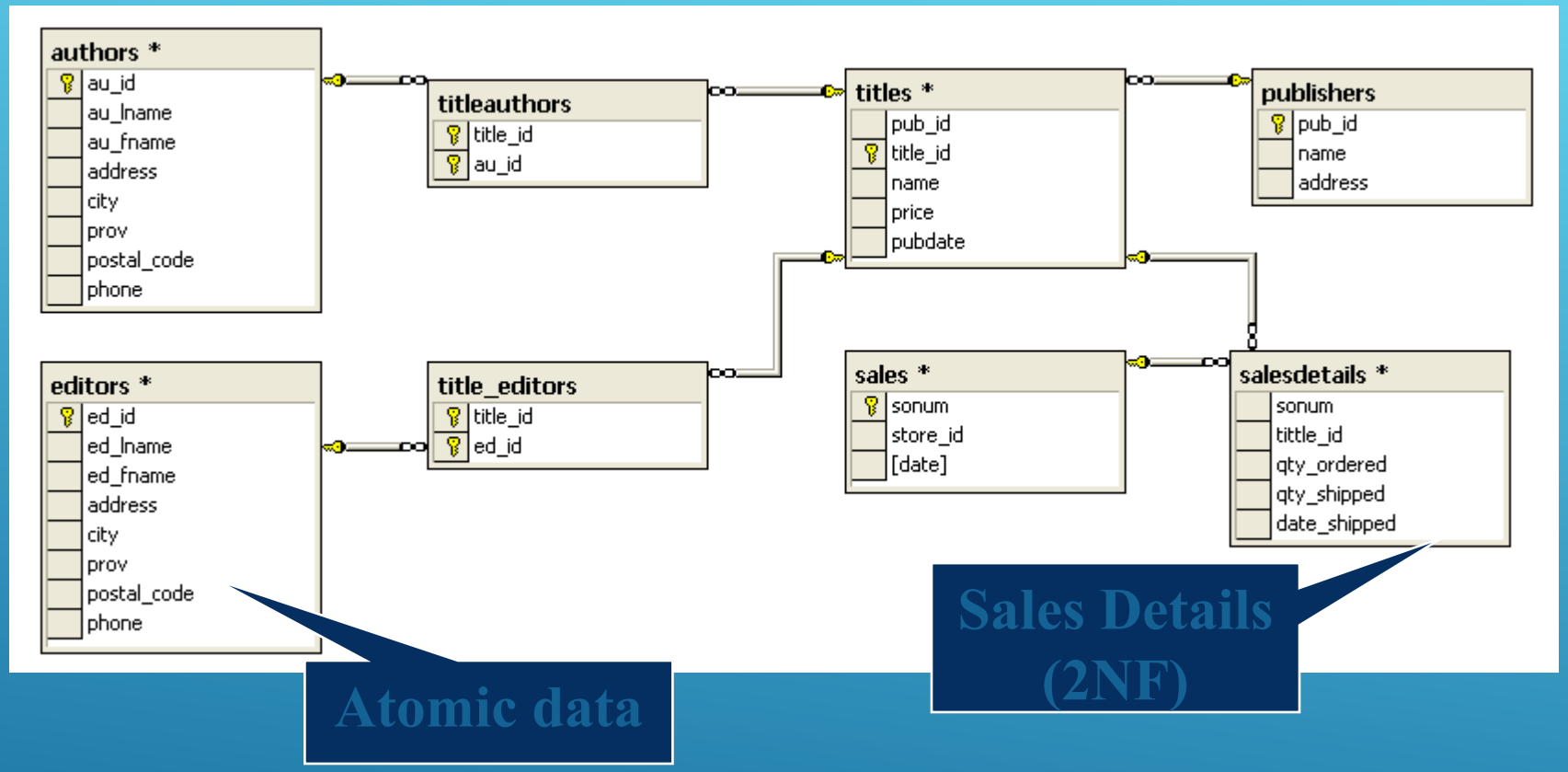
2. Copy the key column(s) that the non-key column was dependant on into the new table; this will be PK for the new table AND also now FK in the original table (have created a PARENT/CHILD, also called One-to-many relationship)

2NF – ELIMINATING PARTIAL DEPENDENCIES

Sonum	Store_id	Date
12342	ED21	2021-02-15
12453	ED15	2021-02-20
12654	ED30	2021-03-01

Sonum	Title_id	Qty_ordered	Qty_shipped	Date_shipped
12342	BU1032	25	20	2021-03-01
12342	RV4135	50	50	2021-03-03
12453	TR2176	100	80	2021-03-01
12453	BU1032	30		
12654	RV4135	80	80	2021-03-05
12654	TR2176	40	35	2021-03-03

So store_id and date are only dependent on Sonum. This is partial dependency



SECOND NORMAL FORM (2NF)

SECOND NORMAL FORM (2NF)

- A more illustrative example of the 2NF is if we consider the following table

Employee Table		
Employee	Shoe Size	Trade
Mike	10	Plumbing, Roofing
Bob	9	Electrical
Joe	12	Plumbing

- This table in 1NF will be:

Employee Table		
Employee	Shoe Size	Trade
Mike	10	Plumbing
Bob	9	Electrical
Joe	12	Plumbing
Mike	10	Roofing

SECOND NORMAL FORM (2NF)

We see in the previous table that the key has to be a composite one, consisting of both the name and the Trade.

However, Shoe size is dependent on only the person, not the trade.

To bring the data to 2NF, we create an additional table as shown in the next slide.

Employee Table	
Employee	Shoe Size
Mike	10
Bob	9
Joe	12

Trade Table	
Employee	Trade
Mike	Plumbing
Bob	Electrical
Joe	Plumbing
Mike	Roofing

SECOND NORMAL FORM (2NF)

THIRD NORMAL FORM (3NF)

No non-key column should depend on another non-key column

THIRD NORMAL FORM (3NF)

- A good example of 3NF is if we consider the following table(https://en.wikipedia.org/wiki/Third_Normal_form)

Tournament winners

Tournament	Year	Winner	Winner's date of birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

- In this table winner's date of birth is dependent on winner

THIRD NORMAL FORM (3NF)

Tournament winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winner's dates of birth

Winner	Date of birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

- You may notice that Winner is Firstname + Lastname. This would have been a 1NF violation! Note that ALL NF's require that ALL previous NF tests are also passed, so the above example demonstrates the 3NF step ONLY – but the resulting data does NOT technically pass 3NF…

CONVERTING TO 3RD NORMAL FORM

- remove the dependent column(s) and place it in a new table
- Copy the non-key column(s) that the other non-key column(s) depended on and place into the new table as the primary key; the matching columns in the original table must now be made FK (to the PK in the new table)

BENEFITS OF NORMALIZATION

GREATER OVERALL DATABASE ORGANIZATION

REDUCTION OF REDUNDANT DATA

DATA CONSISTENCY WITHIN THE DATABASE

A MUCH MORE FLEXIBLE DATABASE DESIGN

A BETTER HANDLE ON DATABASE SECURITY

DRAWBACKS OF NORMALIZATION

Normalization may need to be compromised to meet speed or hardware limitations

A normalized database may be more flexible, but require more CPU effort to create the datasets requested