Tables & Fields

Converting requirements and ERDs
into data definitions

Tables & Fields

Defining Information to be Stored

Steps in developing an application:

- Diagram the process to be automated (ERD)
- Gather user requirements to understand the process and how the data will be used
- Translate entities into tables
- Translate attributes into fields

Tables & Fields

Defining Information to be Stored

Steps in developing an application:

- Translate relationships between entities into relationships between tables

- Normalize the fields in each table

- Convert the user requirements into user interface features that accomplish tasks

Tables & Fields

Translating Entities into Tables

Each entity in the ERD can be translated directly into a database table

- What is a table?
    - Collection of data organized into fields
    - Instances of data organized into records
- Table definitions are based on the Relational Model

Tables & Fields

Translating Entities into Tables

What is the Relational Model?

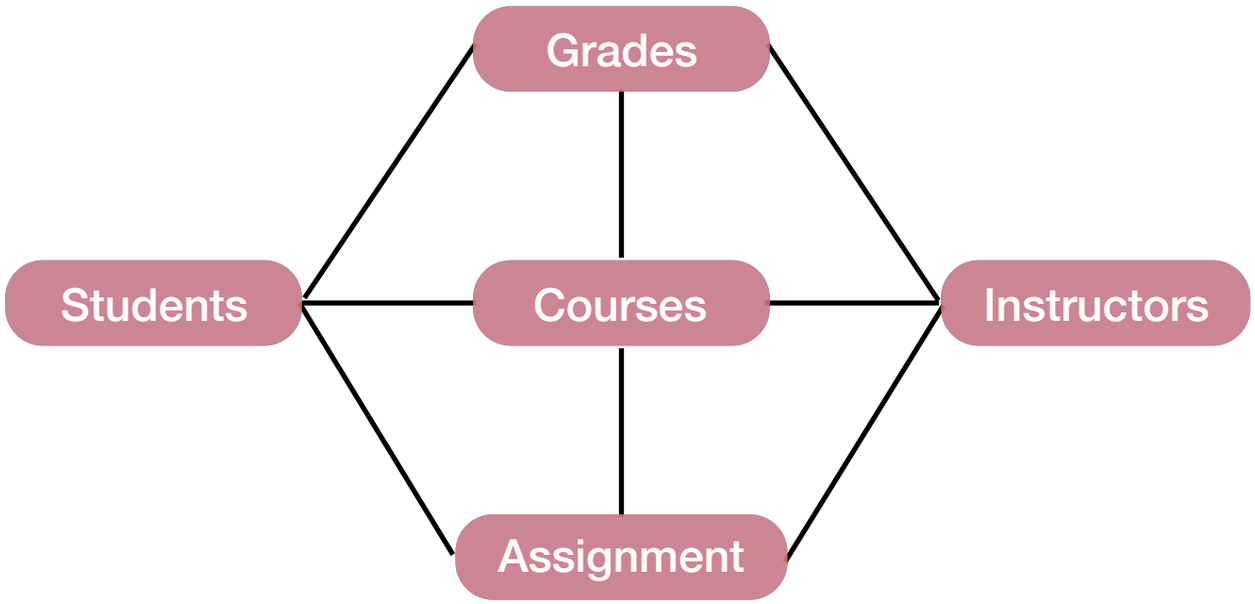
Basic Translation:

- Tables consist of related data (expressed as fields) that when taken together form an instance of an entity/object

- Each record is unrelated to the other records within the same table

Tables & Fields

Translating Entities into Tables



Each Entity Becomes Its Own Table

Tables & Fields

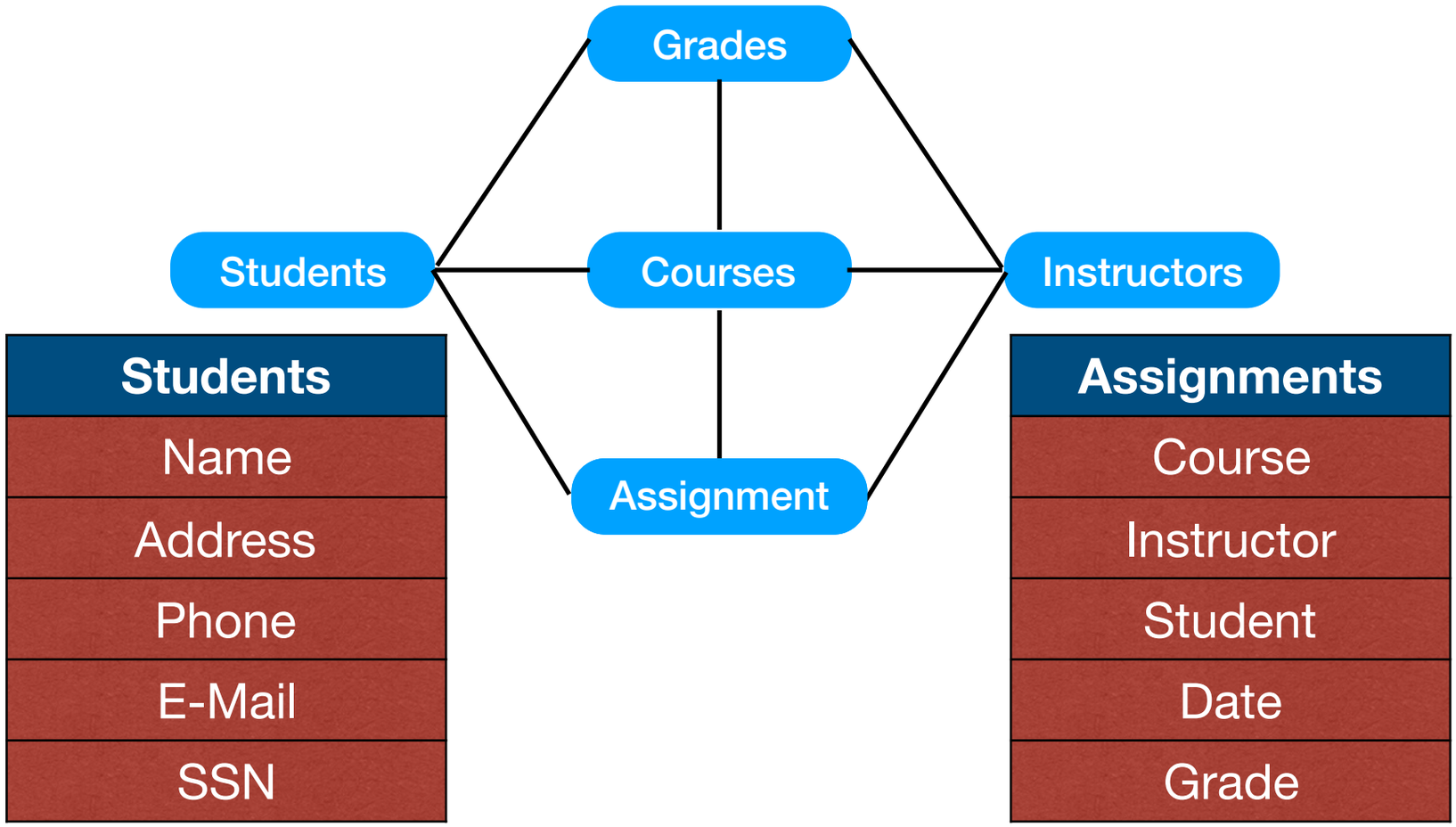
Translating Attributes into Fields

Each attribute of an entity can be translated directly into a database field (aka column)

- What is a field?
    - Data that describes the entity
    - Links one entity to another
- Fields are defined based upon their type

Tables & Fields

Translating Entities into Tables



Tables & Fields

Primary Keys

- Each record must have a method of uniquely identifying it among all other records
- The field that is unique among all records is called the Primary Key
    - Sometimes multiple fields can form a Primary Key, as long as the combination is unique
- If no field is unique, then an ID field should be added to the table

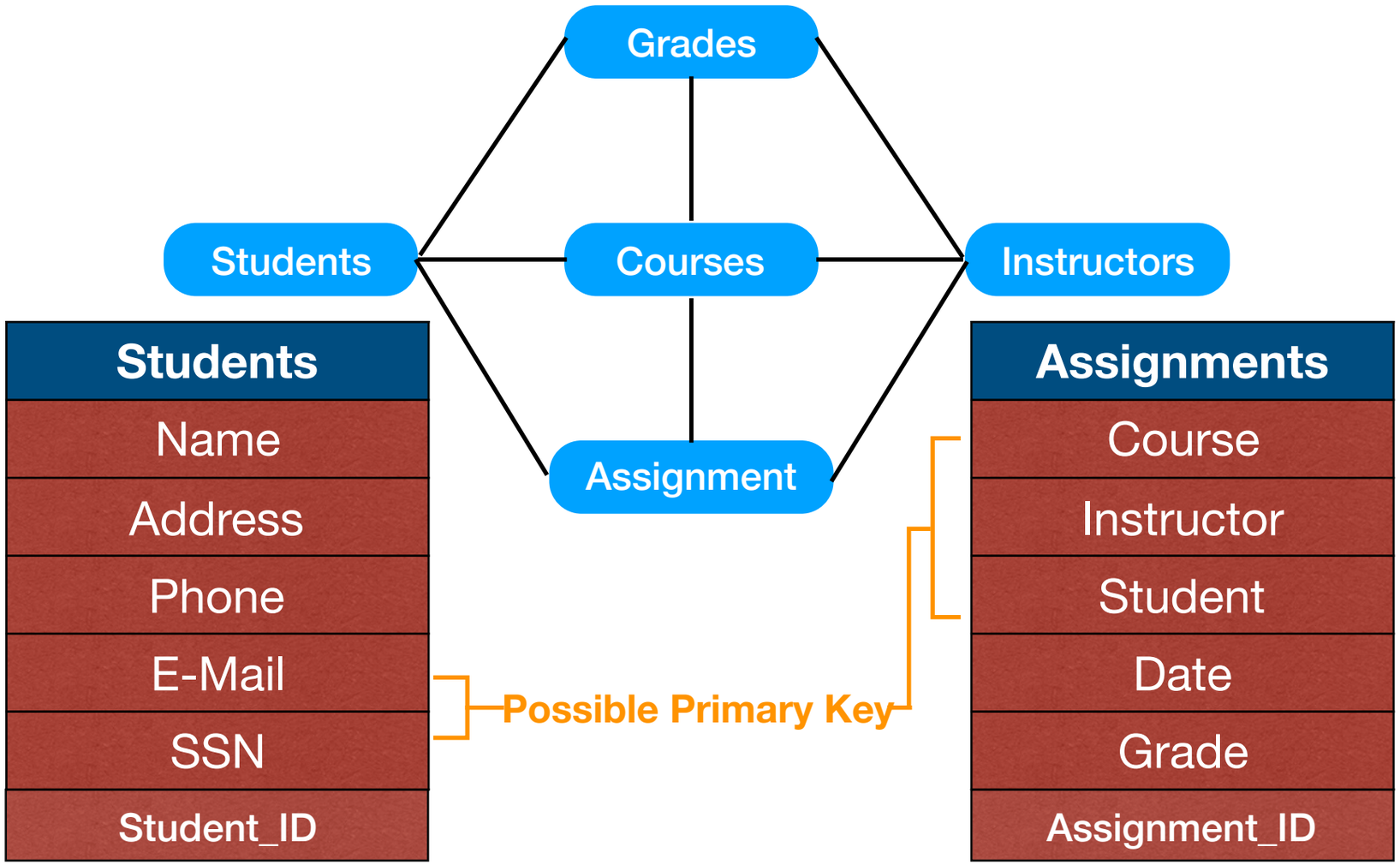
Tables & Fields

Primary Keys

- ID fields are auto-incrementing integers
    - Each successive record gets the next number
    - Its only purpose is to identify the record
- ID fields should be used in related tables also
    - Prevents accidental duplicate primary keys
    - Helps better identify specific records

Tables & Fields

Primary Keys



Relationships

Linking Data Together

Relationships

Relating Data Between Tables

- The ERD showed relationships between entities
- Database tables should include those same relationships
- The verbs used to describe them in the ERD show how to establish them in a database

Relationships

Relating Data Between Tables

- How to establish a relationship between records:
  - One table has a field that contains a unique value in it
  - The other table has a field that contains the same value
- ERD descriptions determine how the relationship is defined between the tables

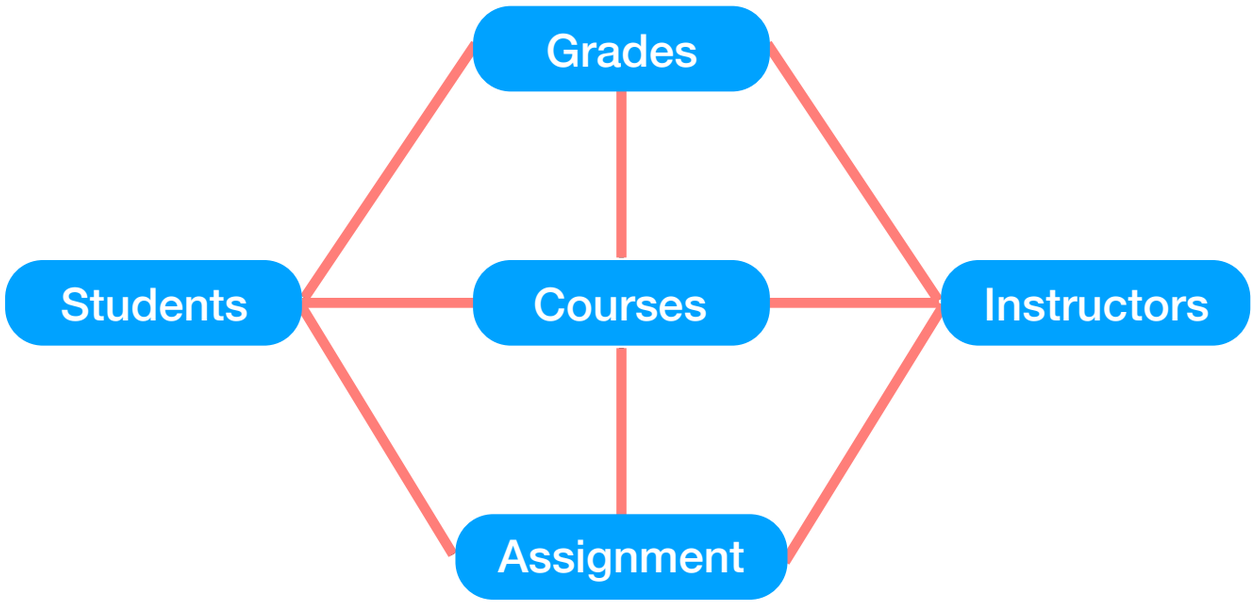
Relationships

Relating Data Between Tables

- Each relationship has a parent table and a child table
- When linking to the parent table, the unique ID field is used
- The child table contains a field that matches the field type of the ID field in the parent

Relationships

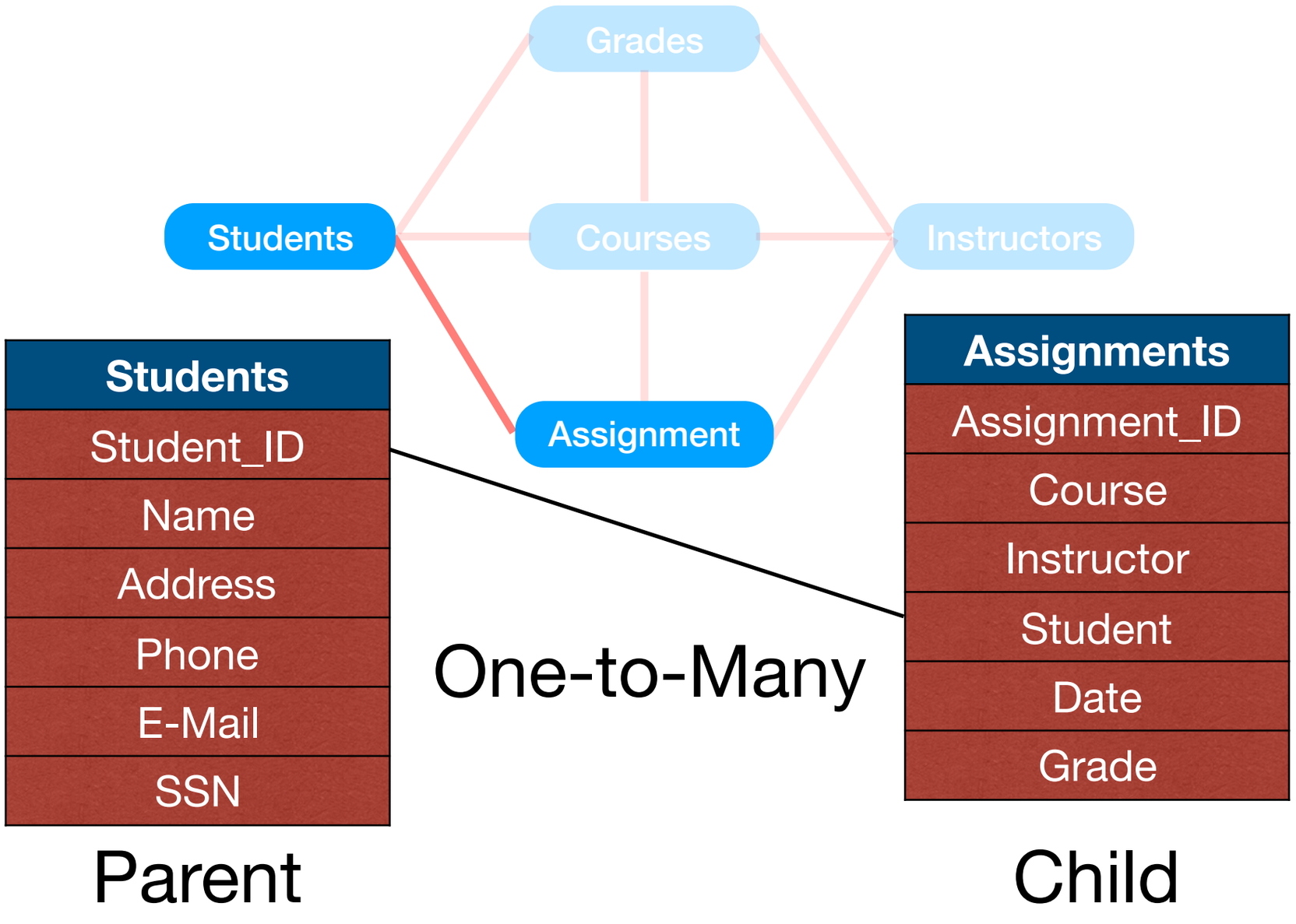
Relating Data Between Tables



Each Relationship is Translated to the Database

Relationships

Relating Data Between Tables



Relationships

Understanding Relationships

- One-to-Many indicates there are specific rules:
    - Only one parent record for each child
        - Any number (including zero) of children records for each parent
- No child can exist without a parent
- Referential integrity rules enforce this setup

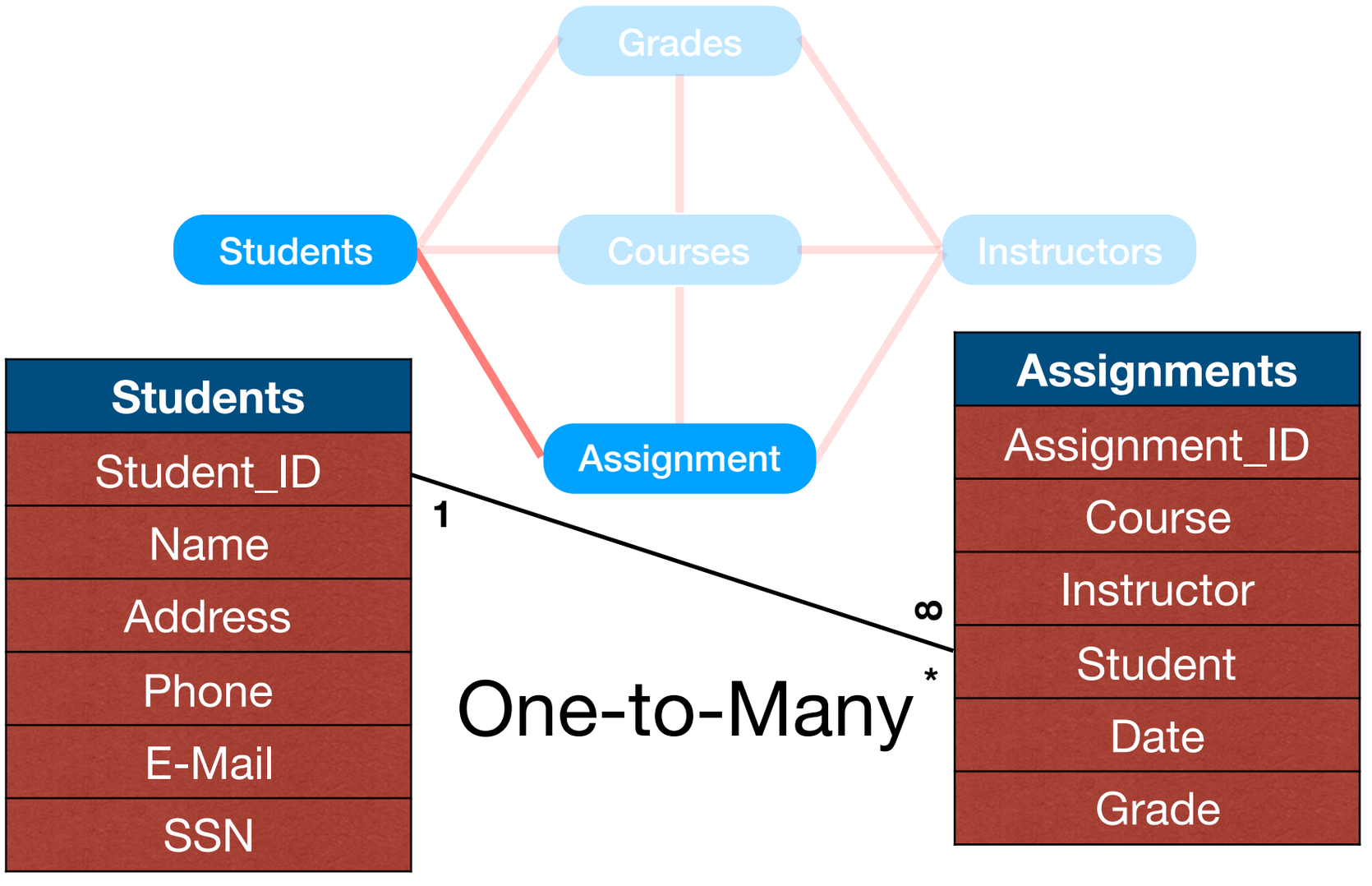
Relationships

Understanding Relationships

- One-to-One indicates a special form of One-to-Many
  - Only one child exists
  - Sometimes there is no child record
- Good for times when the parent record has optional data - helps manage disk space better

Relationships

Relating Data Between Tables



Normalization

Optimizing table structures
to reduce data duplication
and prevent modification errors

Normalization

Why Optimize?

- Intent is to make storage & management as efficient as possible
- Reduce duplication of data
  - Prevent possibility of data not updating properly
  - Increases efficiency of data storage
  - Simplify building of queries

Normalization

Why Optimize?

- Efficiency sometimes results in complexity
- Complexity may seem to hinder the database
    - Additional steps may be needed to store data
    - Queries can appear complicated

Normalization

Classification Levels

- There are six levels of normalization with additional related levels or subsets
- Each level builds off of the previous level
- Each level is called a ‘form’
  - 1st normal form
- Strive to achieve 4th normal form
- Achieving 3rd normal form is deemed as a database that is normalized

Normalization

What is it?

The process of normalizing a database:

- Reduce data into its smallest practical units
- Properly show relationships among data
- Eliminate duplication

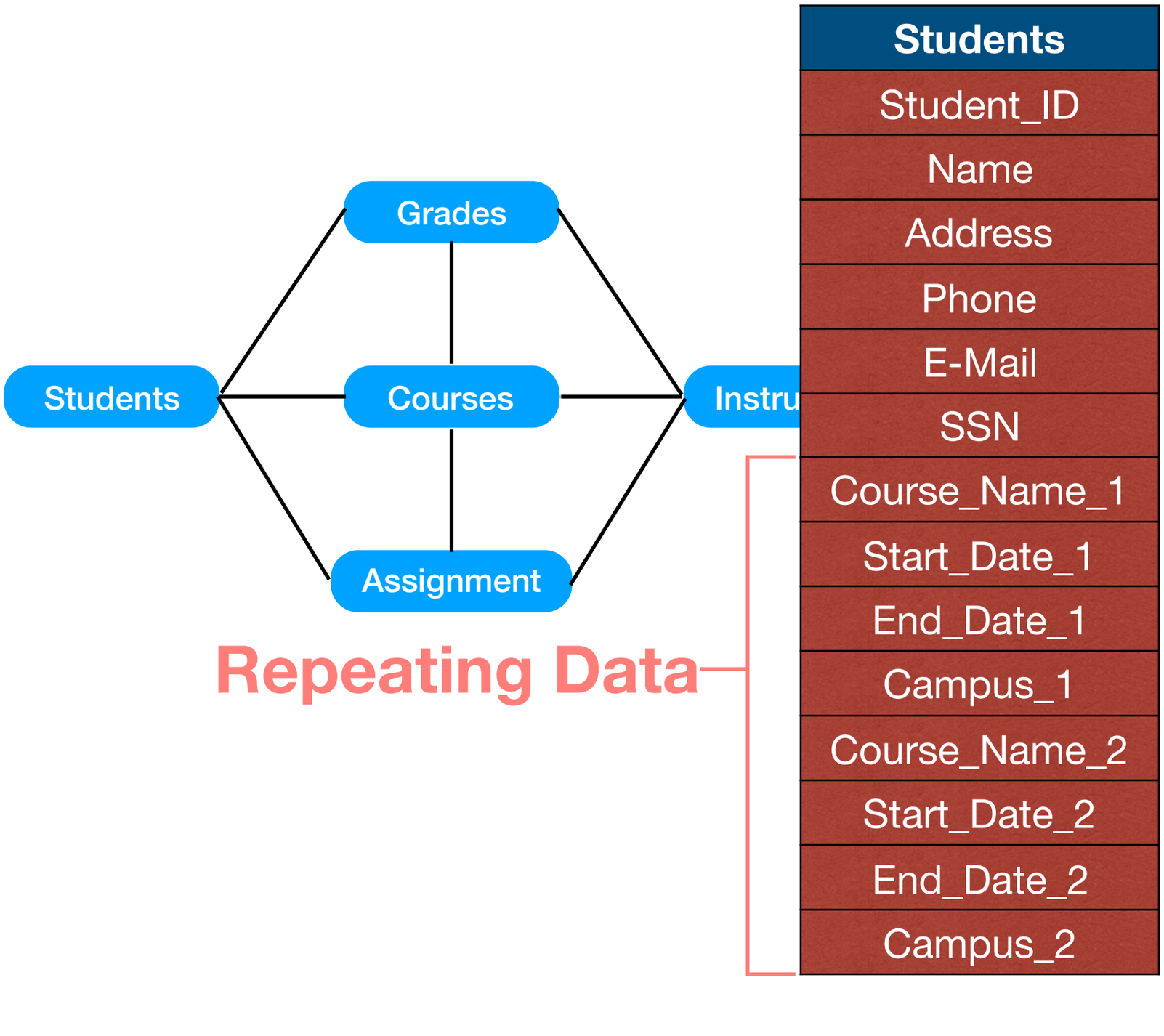
Normalization

1st Normal Form

- Definition: No repeating data types
- Translation: Fields that appear multiple times in a single table should be in their own table and linked to the original

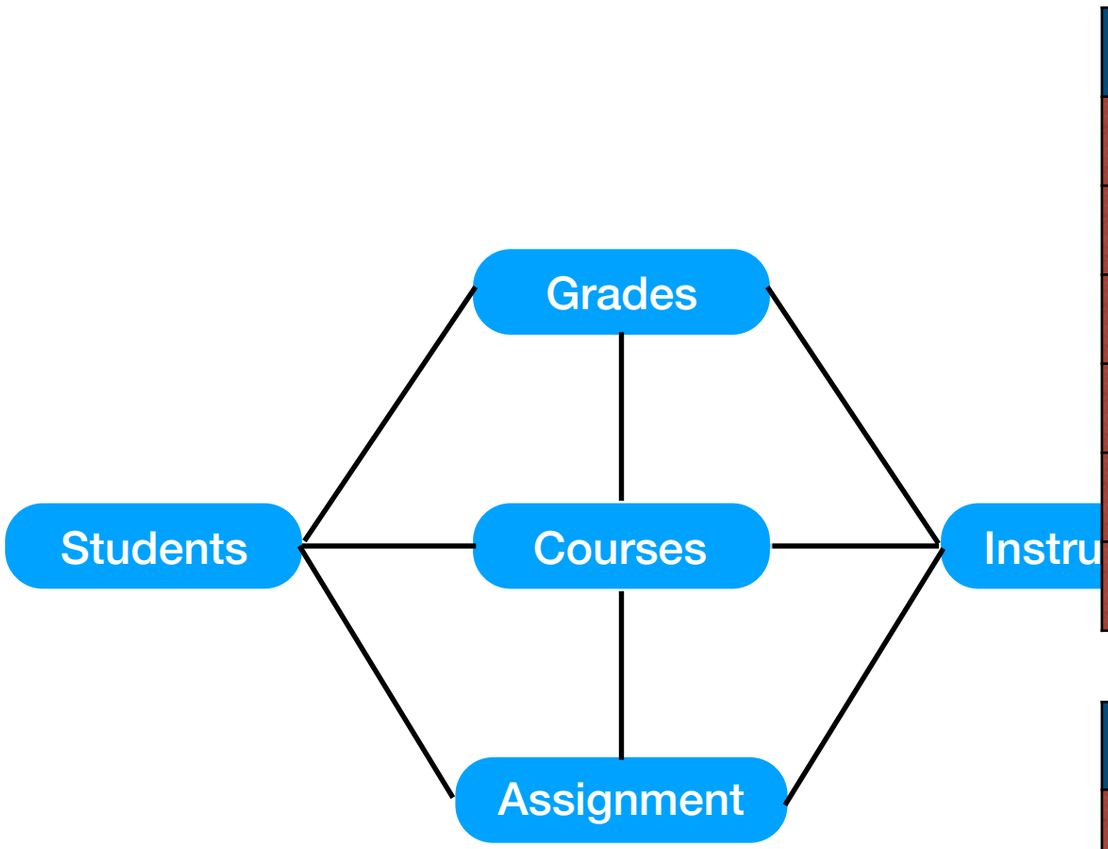
Normalization

1st Normal Form



Normalization

1st Normal Form



Students
Student_ID
Name
Address
Phone
E-Mail
SSN

Enrollments
Student
Course_Name
Start_Date
End_Date
Campus

Normalization

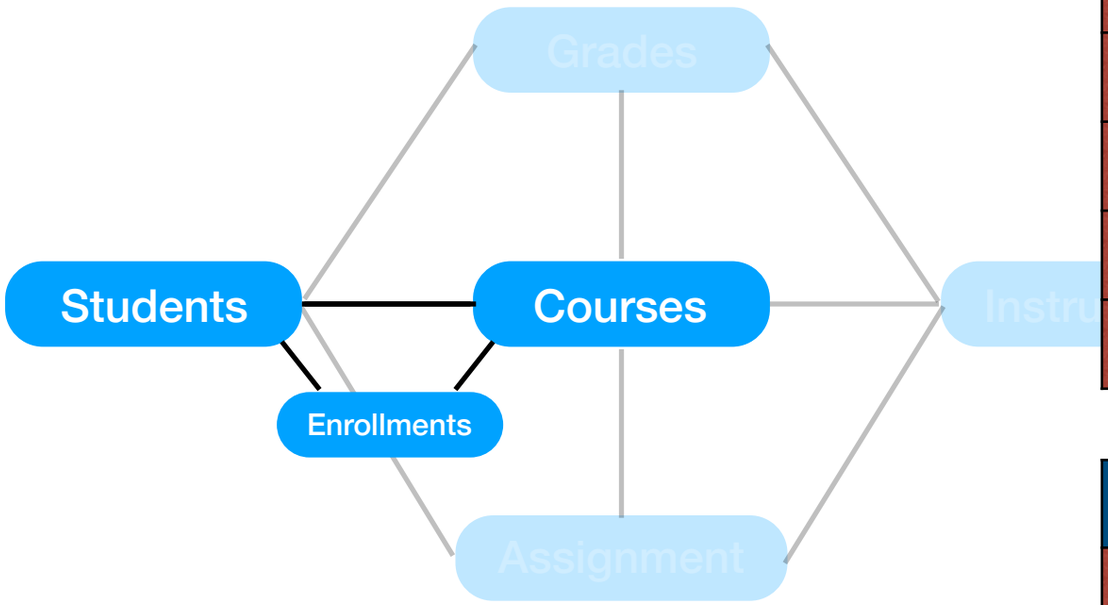
2nd Normal Form

- Definition: All data is uniquely identified by the primary key

- Translation: Information in a table should describe the actual subject of the table and not belong to related material

Normalization

2nd Normal Form



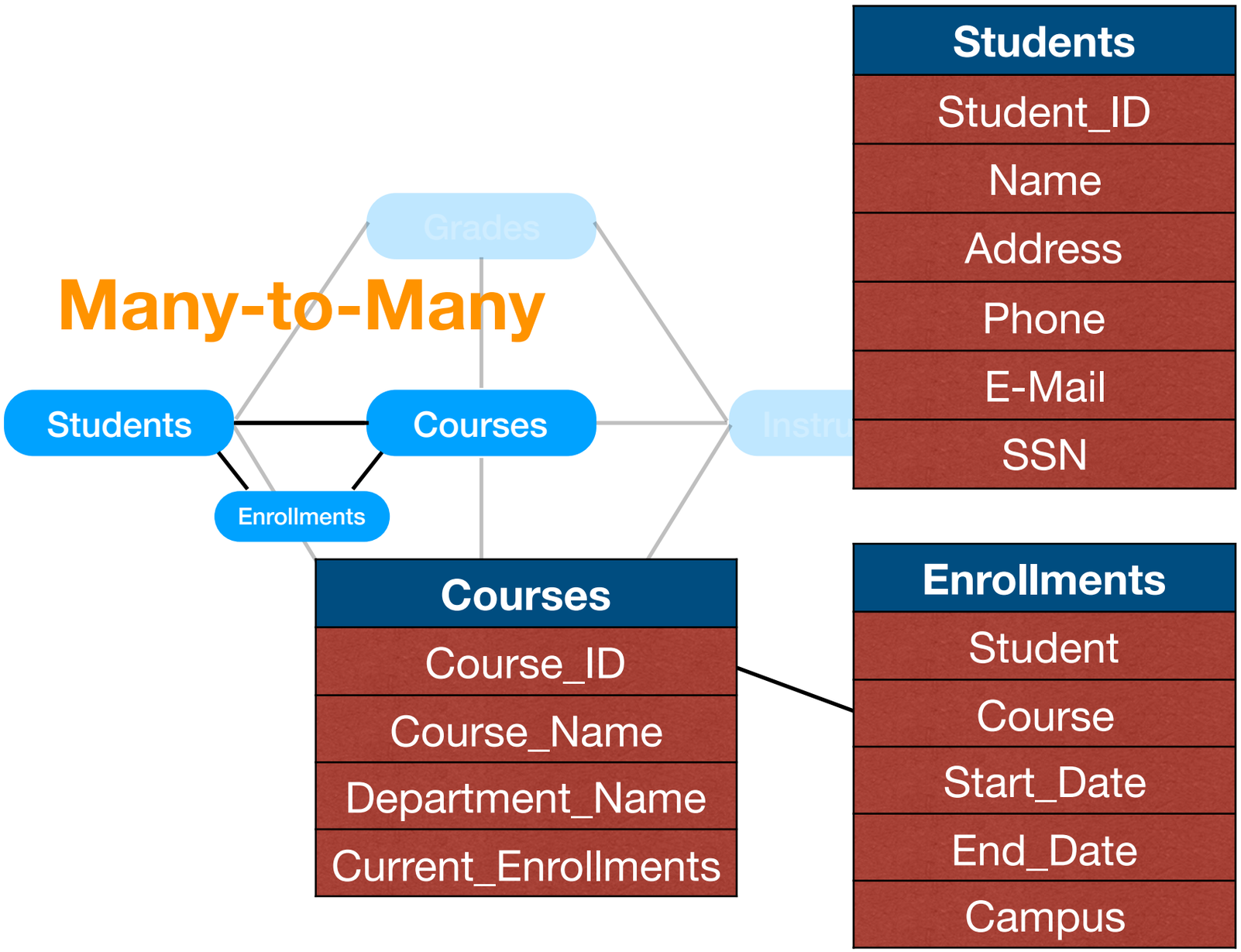
Students
Student_ID
Name
Address
Phone
E-Mail
SSN

Unrelated Data →

Enrollments
Student
Course_Name
Start_Date
End_Date
Campus

Normalization

2nd Normal Form



Normalization

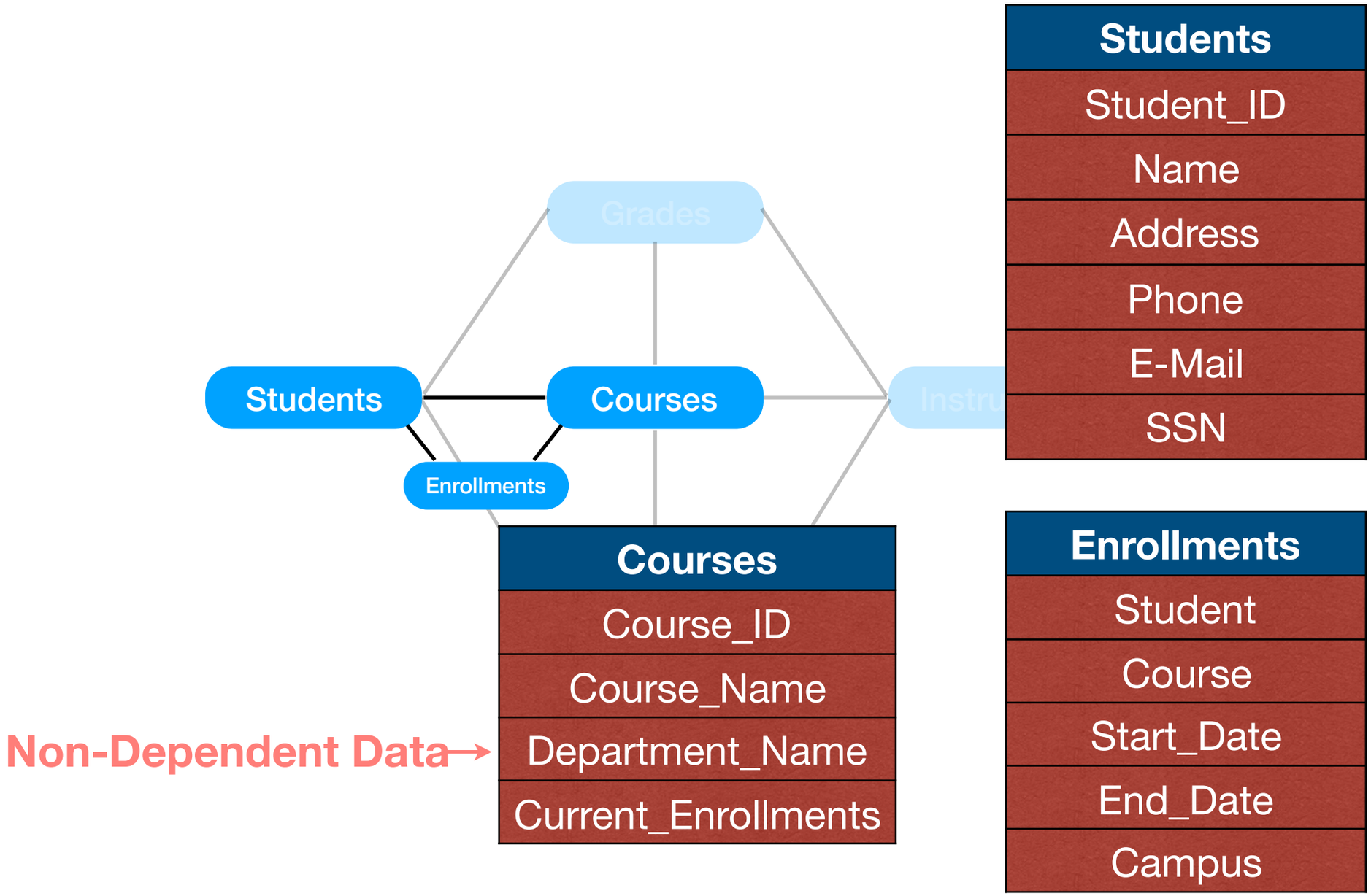
3rd Normal Form

- Definition: Remove fields that aren't dependent on the primary key

- Translation: If a field is associated to the table, but it doesn't describe the table entity, then it should be in its own table

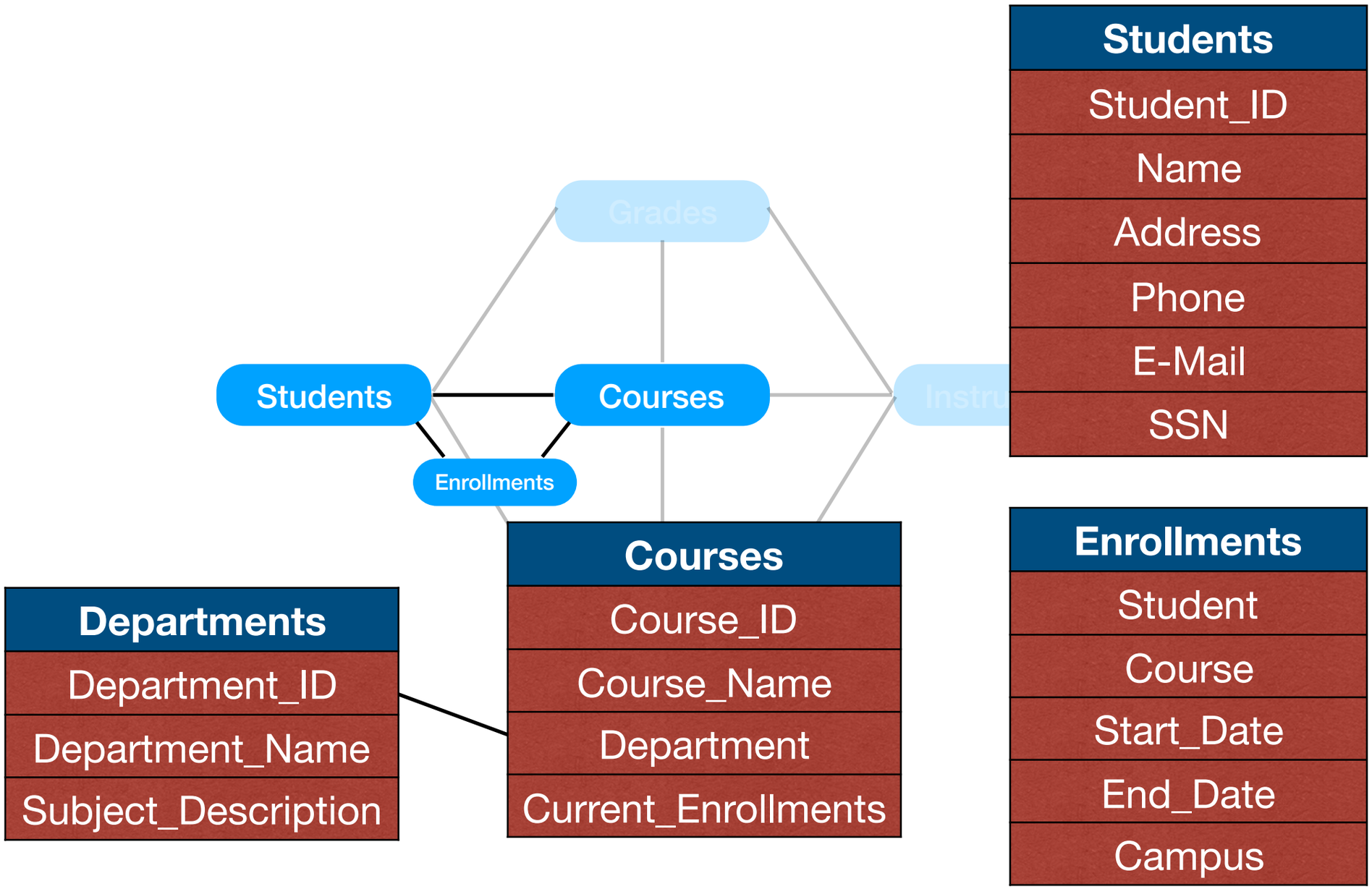
Normalization

3rd Normal Form



Normalization

3rd Normal Form



Normalization

Other Normal Forms

- Boyce-Code Normal Form - If a field in a linking table is dependent upon only one of the linking fields, then it should be removed.

- 4th Normal Form - Don’t combine multiple many-to-many relationships into one table.

Normalization

Other Normal Forms

- 5th Normal Form - If a table can be broken down into multiple related tables, then do it (unless doing so causes joining problems).

- Domain/Key Normal Form - Includes the domain of possible values for a field. A primary key and all domain restrictions must be unique.

- 6th Normal Form - Accounts for temporal nature of data.