1 Data Modeling

1.1 Date and Text Representations

- *Date:* Epoch times since 1970 in seconds. Dates are complex so it's best practice to use language libraries.
- *ASCII:* 8 bits (or 1 byte) per a character. Can represent 127 characters. [1]
- *UTF-8:* The most common text encoding. Can represent up to 1 million characters. Has variable-length encoding; characters are represented by one, two, three, or four bytes. Backward-compatible with ASCII.

1.2 Data Modeling

What data types are appropriate for the following situations?

- Quantity of goods stocked in a warehouse
  - Answer: int
- A Northwestern student ID number
  - Answer: An NU id has 7 digits and we need to be able hold at-least $10,000,000$ numbers, so a int is appropriate.
- A student's name
  - Answer: UTF-8
- Energy readings produced by a particle accelerator
  - Answer: Double since the scientist need precise measurements.
- Temperature in Celsius, recorded to the tenth of degree
  - Answer: fixed point
- A transaction in dollars and cents in a banking application
  - Answer: fixed point, however we need it to be a signed fixed point for inflow/outflows.
- A ten-digit phone number
  - Answer: long (in 64-bit machine). Since we need to store at-least 10 billion different values and an int can only represent 4 billion. We can also use ASCII but then we would require 10 bytes instead of 8 bytes.
- A plain-text password
  - Answer: UTF-8

[1] ASCII Uses 7 bits per character, but in practice each character is stored in 8 bits and the top bit is zero. Hence $2^7 - 1 = 127$

Fundamental data types for integers and floating-point constants [2]

Data Type	32-bit size	64-bit size
char	1 byte	1 byte
short	2 bytes	2 bytes
int	4 bytes	4 bytes
float	4 bytes	4 bytes
long	4 bytes	4/8 bytes
double	8 bytes	8 bytes
long long	8 bytes	8 bytes

[2] Floating-point data types are `float, double, long double.` Integer data types are `char, short, int, long, long long.` Integer types may be prefixed with the `signed` or `unsigned` qualifier. If no sign qualifier is present, the type is assumed to be signed.

2 Relational Databases

2.1 Relational Databases

Database schemas define:

- Tables
- Columns (column names and data types)
- Primary keys
- Foreign keys

Normalization is the process of breaking one redudant table into multiple tables.

2.2 Primary Keys and Foreign Keys

A table's **primary key** uniquely identifies each row. No two rows in a table can have the same primary key. A table's **foreign key(s)** refer to keys in other tables.

2.3 Data Normalization Example

An office supply store is digitizing its customer database. All of their customer data is currently stored on paper order forms.

Order Form

Order number: 1234 Date: 11/04/98

Customer number: 9876

Customer name: Billy

Customer address: 456 HighTower Street

City–Country: Hong Kong, China

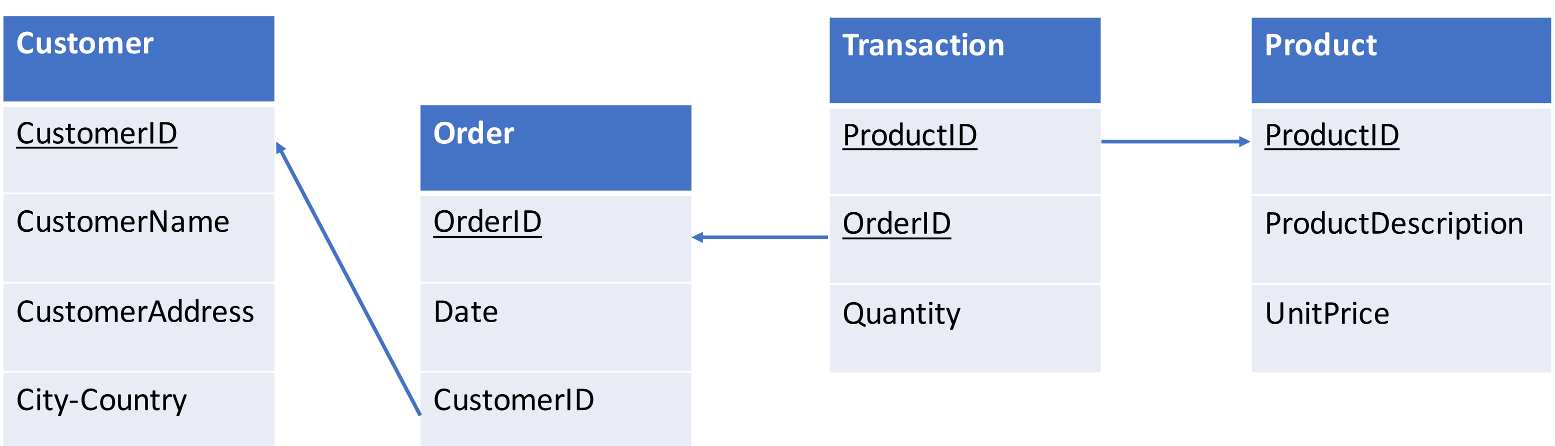
ProductNo	Desscription	Quantity	Unit Price
A123	Pencil	100	$3.00
B234	Eraser	200	$1.50
C345	Sharpener	5	$8.00

What is the widest schema possible that can be used to represent an order?

Widest schema possible

CustomerID
CustomeName
CustomerAddress
City-Country
ProductID
ProductDescription
UnitPrice
OrderID
Quantity
Date

- What are some problems with using such a wide schema?
  - Redundant data is a problem since you must duplicate the customer and product specific information for each order that is received.
  - Updates are also hard. For instance if you have a new product and want to reflect this via data then you would have to update everything not related to the product with `null` values.
- Normalize the database



Columns 1-4 are specific to a customer and `CustomerID` uniquely identifies these items. We call this table **Customer**

Columns 5-7 are specific to a product and `ProductID` uniquely identifies these items. We call this table **Product**. Another reason to create this table is that `ProductID`, `ProductDescription`, and `UnitPrice` can change independent of customer specific information and there is a functional dependence between them (*in general things with functional dependencies belong to the same table.*)

We are now left with columns 8-10: `OrderID`, `Quantity`, and `Date`. Note that `ProductID` and `OrderID` form a composite key and together uniquely identify `Quantity`. `OrderID` alone uniquely identifies `Date`. So create two additional tables:

- **Order**: Consists of `OrderID` and `Date` with primary key `OrderID`.
- **Transaction**: Consists of composite key `ProductID` and `OrderID` along with `Quantity`.

So we normalized the original table into 4 tables: **Customer**, **Product**, **Order** and **Transaction**. Finally **Customer** needs to be linked with **Order** so we create a foreign key `CustomerID` in **Order** to capture this relationship.

Note that `city-country` is a multi-part field. Working with a multi-part field is difficult because its value contains two or more distinct items. It's hard to retrieve information from a multi-part field, and it's hard to sort or group the records in the table by the field's value. To fix this split `city-country` into two fields `city` and `country`.

3 Data Modeling

You are designing a database to manage patient data for a hospital.

- A patient has a name, an age, and a phone number.
- Patients also have a diagnosis, which is a disease.
- Patients can be prescribed to taking drugs with a dosage to treat their illness.
- Each patient has physicians that attend to them. Each physician has a name and a specialty.
- Each physician has a head of department that they need to report to, and a list of residents and nurses that report to them. Residents and nurses also have patients.
- Combinations of drugs can produce side-effects in patients, which are also diseases

Design an ER diagram for this database and be explicit about the relationships in the diagram. For simplilcity, we can use `Employee` to represent physicians, nurses and department heads. See the answer on the next page when finished.

Data Modeling