



THE UNIVERSITY  
*of* ADELAIDE

# Web and Database Computing •

**adelaide.edu.au**

Intro to Relational Databases: Relational Database Schemas

# Database Schema

The schema defines the relational model for a database; how data is divided into tables in Database design has an impact on storage requirements and efficiency of accessing data.

Good schema design delivers:

- Minimal redundancy of information
  - The same information should not appear in multiple places
- Easy to understand the relationship of the data
  - Information is properly organised or split into logical pieces it easy to access.
- Database performs fast and efficiently
  - Data is organised to find information, with fewer steps, and without retrieving more data than needed.

Important because most web applications are transaction intensive, i.e., users are often creating, reading, updating, and deleting data.

**But how do we design a Schema?**

# Designing Databases

**Step 1:** Specify/Understand requirements

**Step 2:** Model the data (optional - we'll skip this)

**Step 3:** Develop a relation schema **Step 4:** Implement relation schema

# Step 1: Specifying/Understanding Requirements

Defining data requirements needs an understanding of how customers or users will use the data

Things to consider

- What data is available?
- How the data is related?

**Let's look at our previous example**

# Step 1: Specifying/Understanding Requirements

The store wants to keep a record of all of their customers and the items they've purchased during each shopping trip.

## What data do we need?

- Customer Details
- Items Purchased
- Total cost of items
- Shopping trip Date/time
- Store shopped at

## How the data is related?

- Customers make shopping trips
- Items are purchased on Shopping Trips

# Step 1: Specifying/Understanding Requirements

Break this data down into specifics/detail

- Customer Details
  - Customer Name
  - Customer Phone Number
  - Customer Address
  - Customer Loyalty Card No ?
- Items Purchased
  - Name
  - Description
  - Barcode/Inventory number
  - Price
- Cost of items
- Shopping trip Date/time
- Store shopped at
  - Store Address

Now that we know what information we need to store, let's build it into a database...

# Step 2: Model data using high-level (ER) model

In large/complex systems, before we create our schema, we develop a high-level model of the data in the system, and how it is related.

An Entity-Relationship (ER) model illustrates the relationship of the data and can be easily derived into a database schema, reducing the likelihood of errors.

In this course, the systems we're building are simple enough that we don't need ER Models; Instead, we're going to focus on building a robust schema.

You can learn more about database modelling and different types of Databases in [COMP SCI 2208 Databases and Ethical Data](#)



# Step 3: Develop a relation schema

A database schema is a blueprint for the structure of our database.

Consists of **tables** organised into **columns**, representing groups of **attributes** that each **entity/row** in the table shares.

- Each row must be **unique**.
  - The column(s) that make a row unique are called the **Primary Key**
  - Every table must have a Primary Key
- Tables can be related to each other
  - This relationship is established by a reference to the other table's Primary Key
  - This is called a **Foreign Key**

This can be illustrated through a **Schema Diagram**

## Step 3: Develop a relation schema

# Group Related Data, Remove Unnecessary Data

- Customer Details
  - Customer Name
  - Customer Phone Number
  - Customer Address
  - Customer Loyalty Card No ?
- Items Purchased
  - Name
  - Description
  - Barcode/Inventory number
  - Price
- Cost of items
- Shopping trip Date/time
- Store shopped at
  - Store Address

## Step 3: Develop a relation schema

# Group Related Data, Remove Unnecessary Data

- Customer
  - Name
  - Phone Number
  - Address
  - Loyalty Card No
- Item
  - Name
  - Description
  - Barcode/Inventory number
  - Price
- ~~Cost of items~~
- Shopping trip Date/time
- ~~Store shopped at~~
  - ~~Store Address~~




Customers
Name
Phone Number
Address
Loyalty Card No

Shopping Trips
Time

Items
Name
Description
Barcode
Price

# Step 3: Develop a relation schema

## Select/Add Primary Keys

- Customers
  -  Customer ID
  - Name
  - Phone Number
  - Address
  - Loyalty Card No
- Items
  -  Barcode/Inventory number
  - Name
  - Description
  - Price
- Shopping Trips
  -  Trip ID
  - Date/Time

Customers
* ID
Name
Phone Number
Address
Loyalty Card No

Shopping Trips
* ID
Time

Items
* Barcode
Name
Description
Price

Is there a column (or combination) suitable for use as primary key? If not, or in doubt, create one.

## Step 3: Develop a relation schema

# Build Relationships

### How the data is related?

- Customers make shopping trips
- Items are purchased on Shopping Trips

So we need to create relationships between

- Customers and Shopping Trips
- Items and Shopping Trips

**Establish a reference to the other table's Primary Key**



## Step 3: Develop a relation schema

# Relationship Cardinality

We want to establish a reference to the other table's Primary Key,  
**But how do we know which table to add the reference to?**

Relationship cardinality describes for each row of a table, how many rows of the related table it can be associated with. Can a customer make multiple shopping trips, can a shopping trip have many customers?

### Cardinality Types

- **1-1 (one to one)**  
e.g. Each customer can make a single shopping trip, and each shopping trip can only be made by a single customer.  
Put the reference on **either** table; **customer** or **shopping trip**
- **1-N (one to many)**  
e.g. Each customer can make many shopping trips, but each shopping trip can only be made by a single customer.  
Put the reference on the **many** table; **shopping trip**
- **N-N (many to many)**  
e.g. Each customer can make many shopping trips, and each shopping trip can have many customers.  
**Doesn't work!** We'll need to change this to **1-N**, but we'll come back to that ...

## Step 3: Develop a relation schema

# Improving our Schema

### Having a robust schema is important!

- A robust schema makes finding and updating data easy.
- A robust schema makes database transactions fast, even for very complex transactions.
- A robust schema prevents data from becoming inconsistent or corrupt.

### How do we ensure our schema is robust?

## Step 3: Develop a relation schema

# Normalisation and Normal Forms

Database normalization is the process of structuring a database, usually a relational database, in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. Codd as part of his relational model.

- [https://en.wikipedia.org/wiki/Database\\_normalization](https://en.wikipedia.org/wiki/Database_normalization)

Normalising our database:

- Reduces redundancy; ensure the same data doesn't appear in multiple places in the database
- Enforces dependencies & integrity; ensure that updating data in the database doesn't break anything
- Can be done in steps, known as normal forms
- There are 6 Normal forms (plus several additional forms between)
  - We want to make sure our schema is **at least 3rd normal form**.



## Step 3: Develop a relation schema

# 1st Normal Form

The domain of an attribute/column must include only atomic (simple, indivisible) values and the value of any attribute in a row/tuple must be a single value from the domain of that attribute.




What this means:

- Each column must only contain 1 value
- That value should not be able to be sub-divided into smaller components

What does this look like for our Schema?

# Step 3: Develop a relation schema

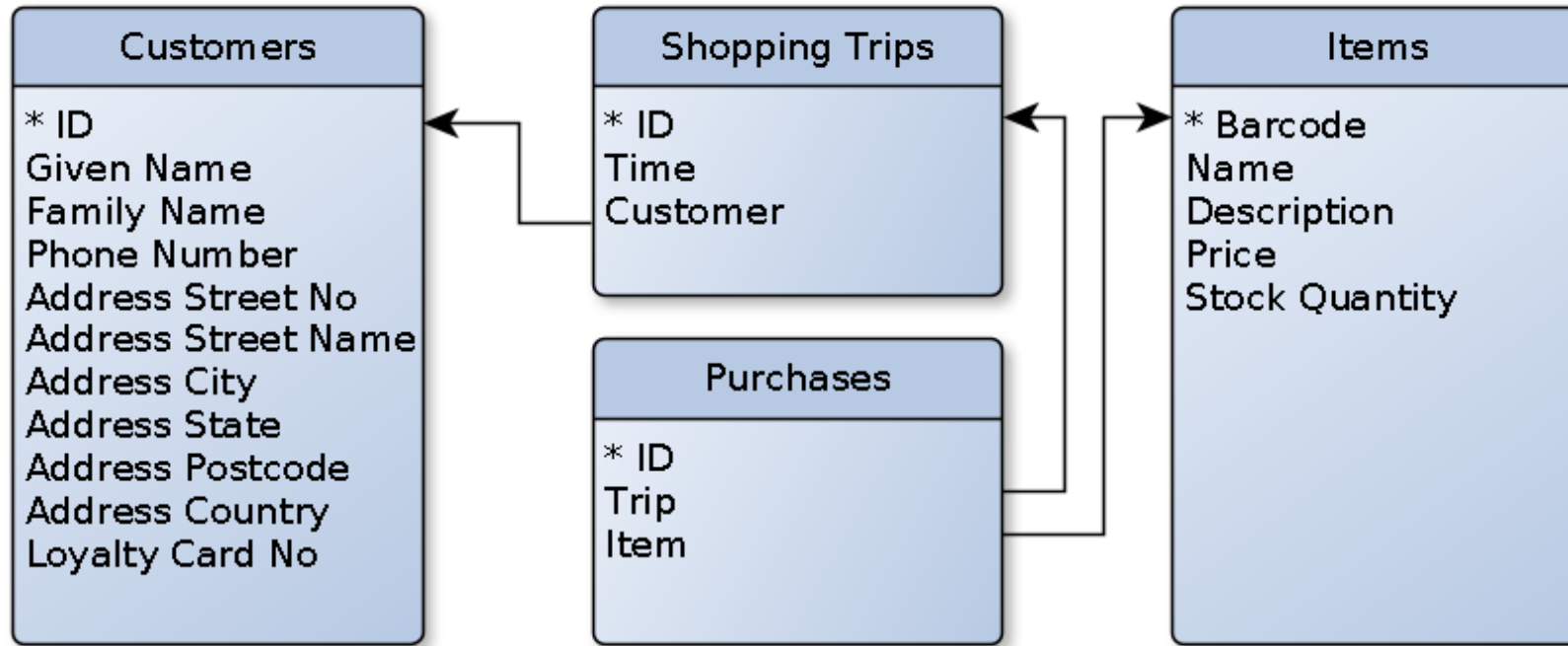
## 1st Normal Form

- Customers
  -  Customer ID
  - **Name**; Can be divided into Given Name, Family Name
  - Phone Number
  - **Address**; Can be divided into Street Number, Street Name, Suburb ...
  - Loyalty Card No
- Items
  -  Barcode/Inventory number
  - Name
  - Description
  - Price
- Shopping Trips
  -  Trip ID
  - Date/Time
  - Customer
  - **Item**; Each shopping trip has multiple items

Need extra columns or tables for these!

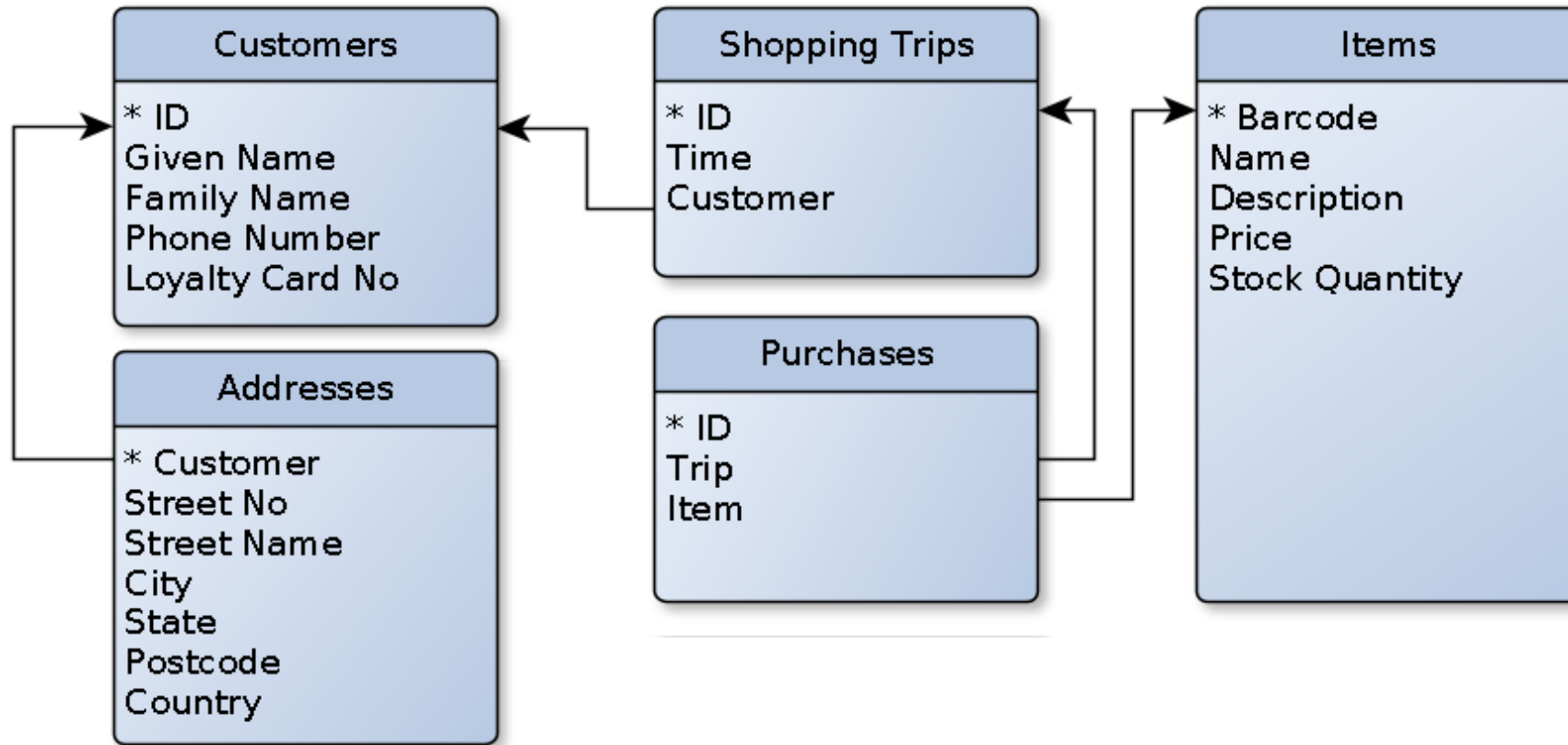
## Step 3: Develop a relation schema

### 1st Normal Form



## Step 3: Develop a relation schema

### 1st Normal Form



## Step 3: Develop a relation schema

# 2nd Normal Form

The schema is in 1st normal form and does not have any attribute/column that is only dependant on part of the key.

What this means:

- When a composite key is used (a primary key consisting of multiple attributes), the other attributes should relate to all parts of the key.

What does this look like for our Schema?

Consider the Purchases table:

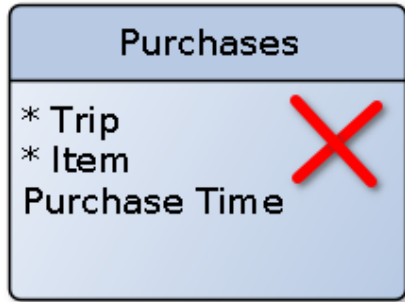
Purchases	
* ID	
Trip	
Item	

## Step 3: Develop a relation schema

# 2nd Normal Form

Assuming each combination of Trip and Item was unique, we could represent this as a composite key.

Consider what would happen if we then added a purchase date field:



This would violate 2nd Normal form because the purchase date is associated with the Shopping Trip, but not **both** the Trip and Item.

In this case, we would either need to create a new table just for Trip and Purchase date, or remove the redundant information if it already existed (it does in the Shopping Trip table.)

## Step 3: Develop a relation schema

### 3rd Normal Form

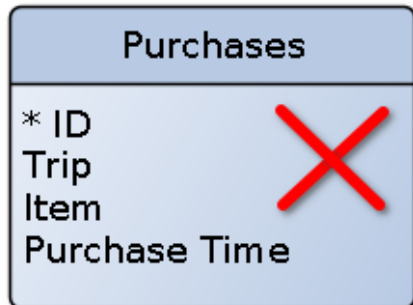
The schema is in 2nd normal form and does not have any attribute/column that dependant on another non-key attribute.

What this means:

- Columns should not reference other columns that aren't keys.

What does this look like for our Schema?

Consider the previous example, but keeping the unique ID:



This would violate 3rd Normal form because the purchase date is associated with the Shopping Trip, and the shopping trip is **not part of the key**.

Again In this case, we would either need to create a new table just for Trip and Purchase date, or remove the redundant information.

## Step 3: Develop a relation schema

# 4th Normal Form

The schema is in 3rd normal form and does not combine many to many relationships in a single table.

What this means:

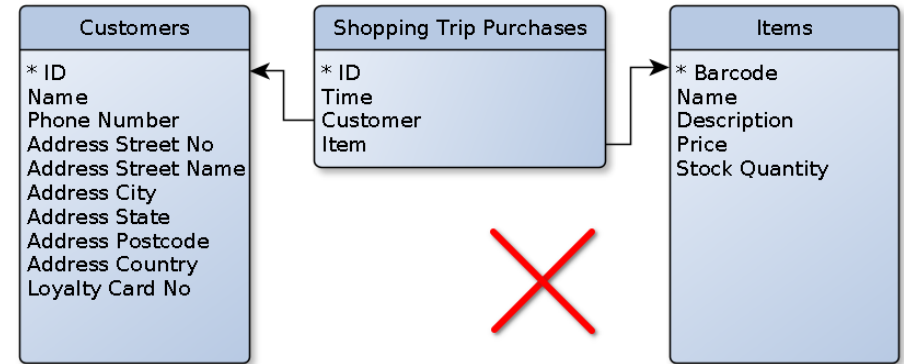
- Do not combine many to many relationships into a single table

Consider if the same shopping trip could be made by multiple users; we would require an extra table for Users on each trip.

But we already have a table for trip & items (purchases).

What if we tried to combine the new table with purchases?

This would violate 4th Normal form because we are essentially using the table to store 2 different record types.





# Once our database is normalised, we're ready to build it!

More resources on Normal Forms & Schema Design:

- A humble guide to database schema design  
<https://www.mikealche.com/software-development/a-humble-guide-to-database-schema-design>
- A Simple Guide to Five Normal Forms in Relational Database Theory  
<http://www.bkent.net/Doc/simple5.htm>
- Database normalization  
[https://en.wikipedia.org/wiki/Database\\_normalization](https://en.wikipedia.org/wiki/Database_normalization)



THE UNIVERSITY  
*of* ADELAIDE

CRICOS PROVIDER NUMBER 00123M