Dr Simon D'Alfonso

INFO 90002
Database Systems & Information Modelling

Week 02

Implementing a Database

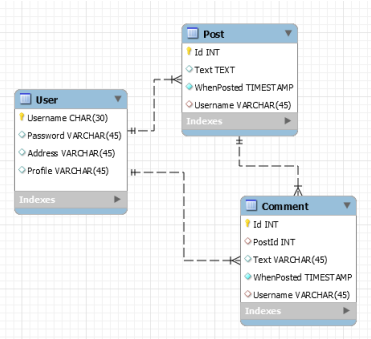
Structure of today's lecture

- Last week: **Designing Databases**
  - homework: noun-verb analysis
  - the database life-cycle
  - modelling a database for an example business
    - conceptual model
    - logical model
    - physical model
- This hour: **Implementing and Using Databases**
  - create the database and tables
  - populate tables with data
  - query data
  - change data

© The University of Melbourne 2018

Database Development Lifecycle

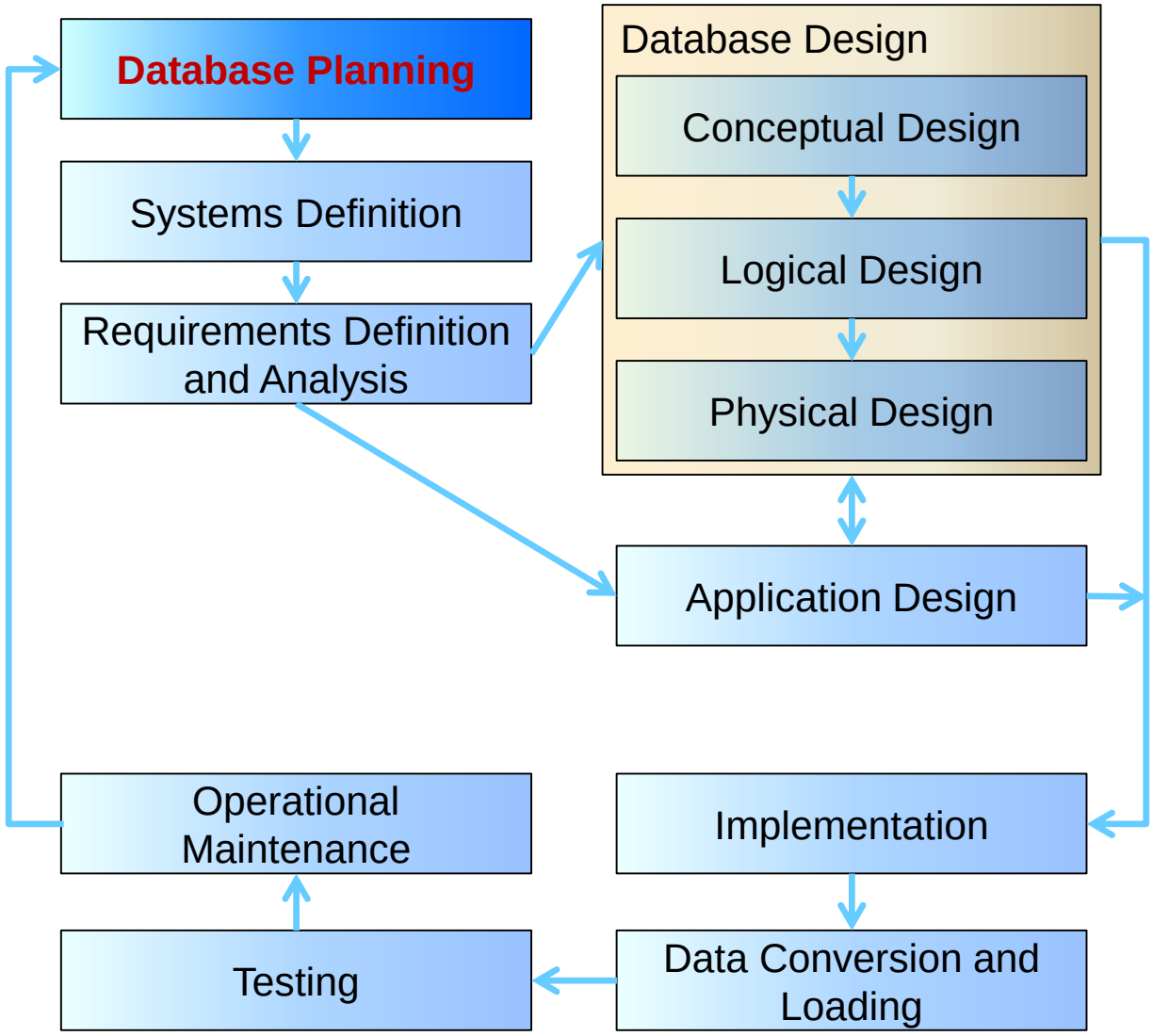
Recap week 1: Database lifecycle



- Design the database
  - data modelling, E-R diagrams
- Implement the database
  - data definition language (DDL)
    - Create
    - Drop
    - Alter
    - Rename
- Data access / programming
  - data manipulation language (DML)
    - Select
    - Insert
    - Update
    - Delete
- Database administration
  - data control language (DCL)
    - Grant
    - Revoke

© The University of Melbourne 2018

Database Development Lifecycle

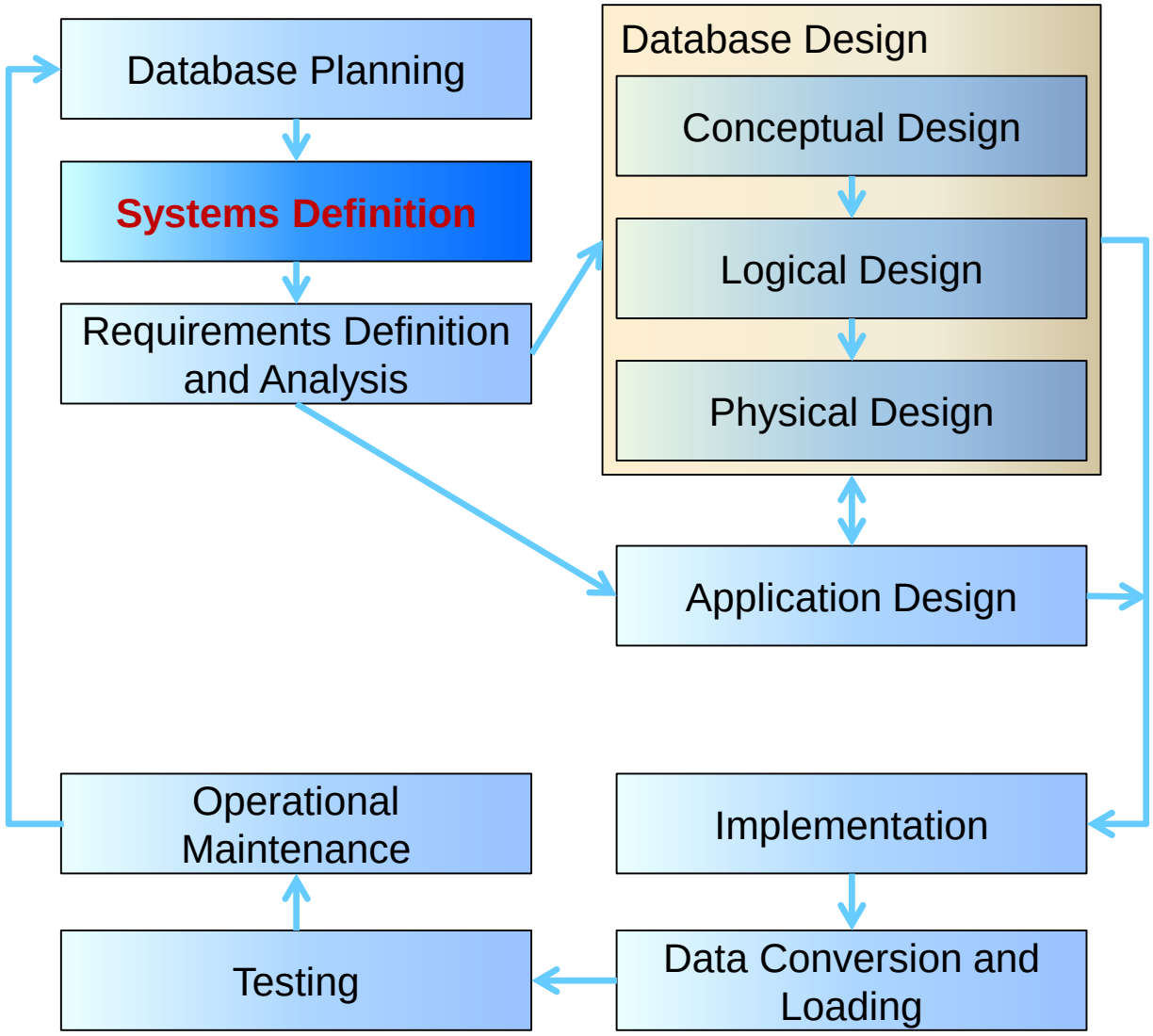


- Planning how to do the project.
  - How does the enterprise work
  - Enterprise data model
- How can the stages be completed efficiently and effectively.

- Outside scope of the course

© The University of Melbourne 2018

Database Development Lifecycle

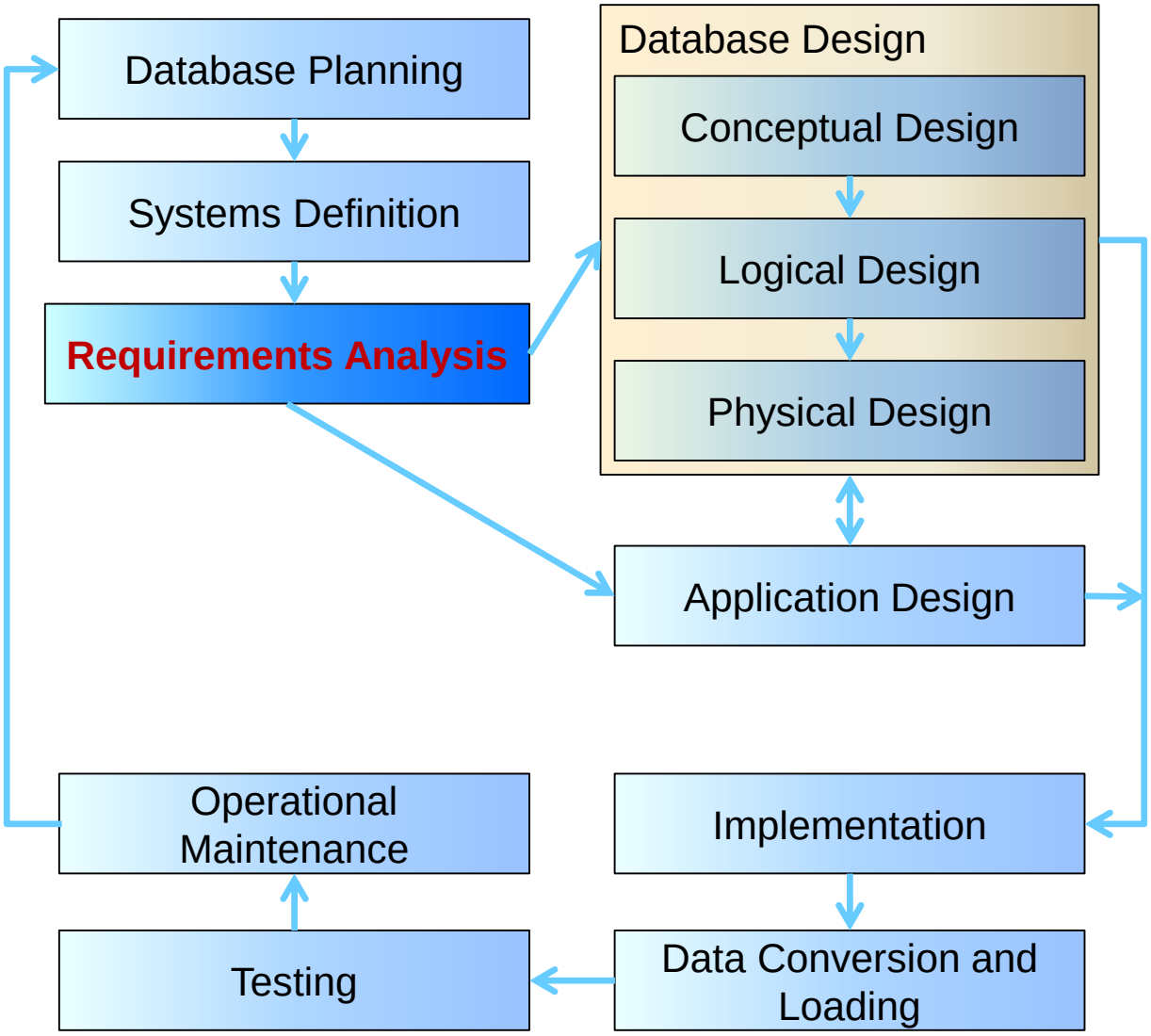


- Specifying scope and boundaries
  - Users
  - Major user views
  - Application areas
- How does it interact with other systems
- User views – how the system operates from differing perspectives

- Outside scope of the course (slightly)

© The University of Melbourne 2018

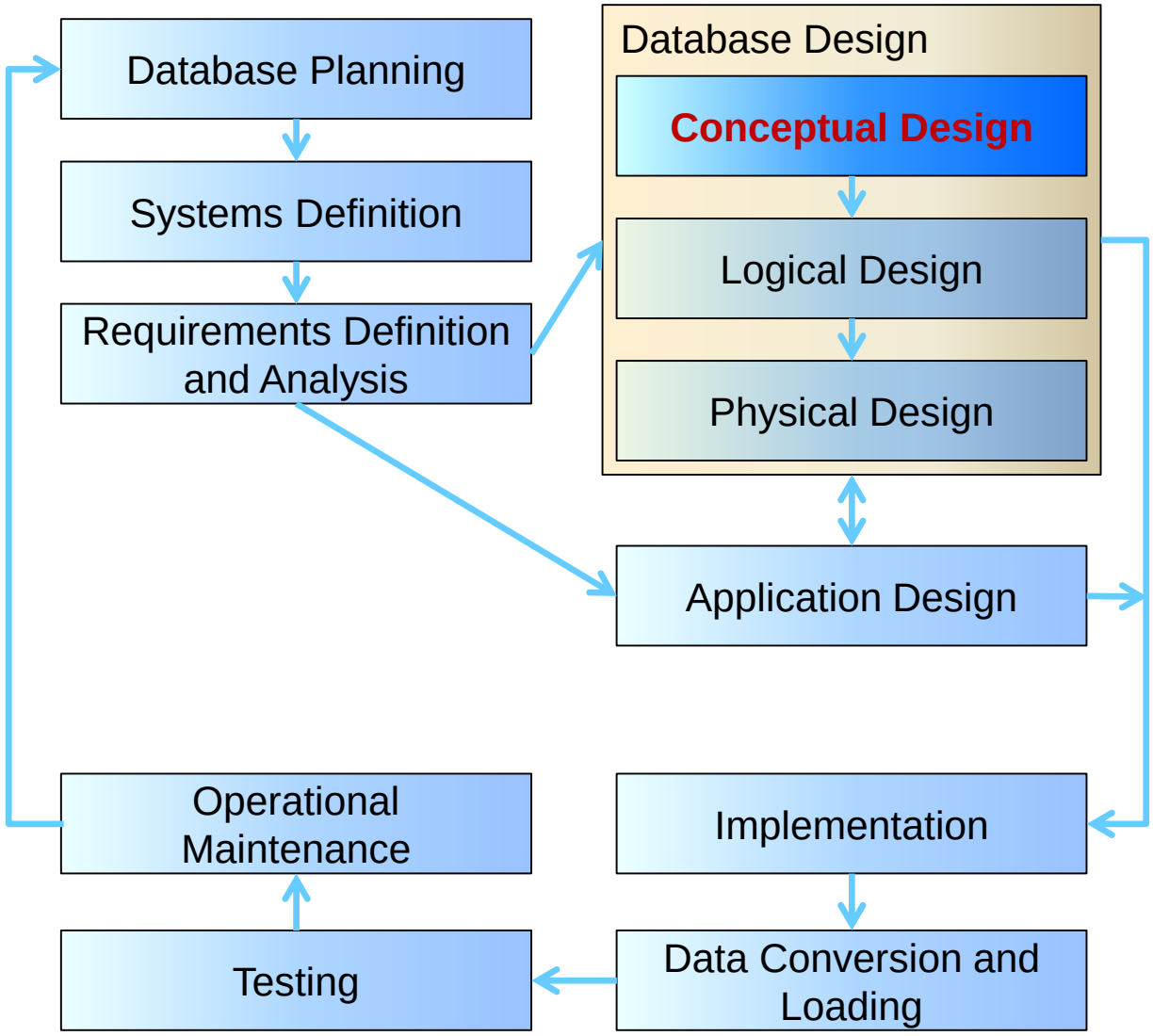
Database Development Lifecycle



- Collection and analysis of requirements for the new system

© The University of Melbourne 2018

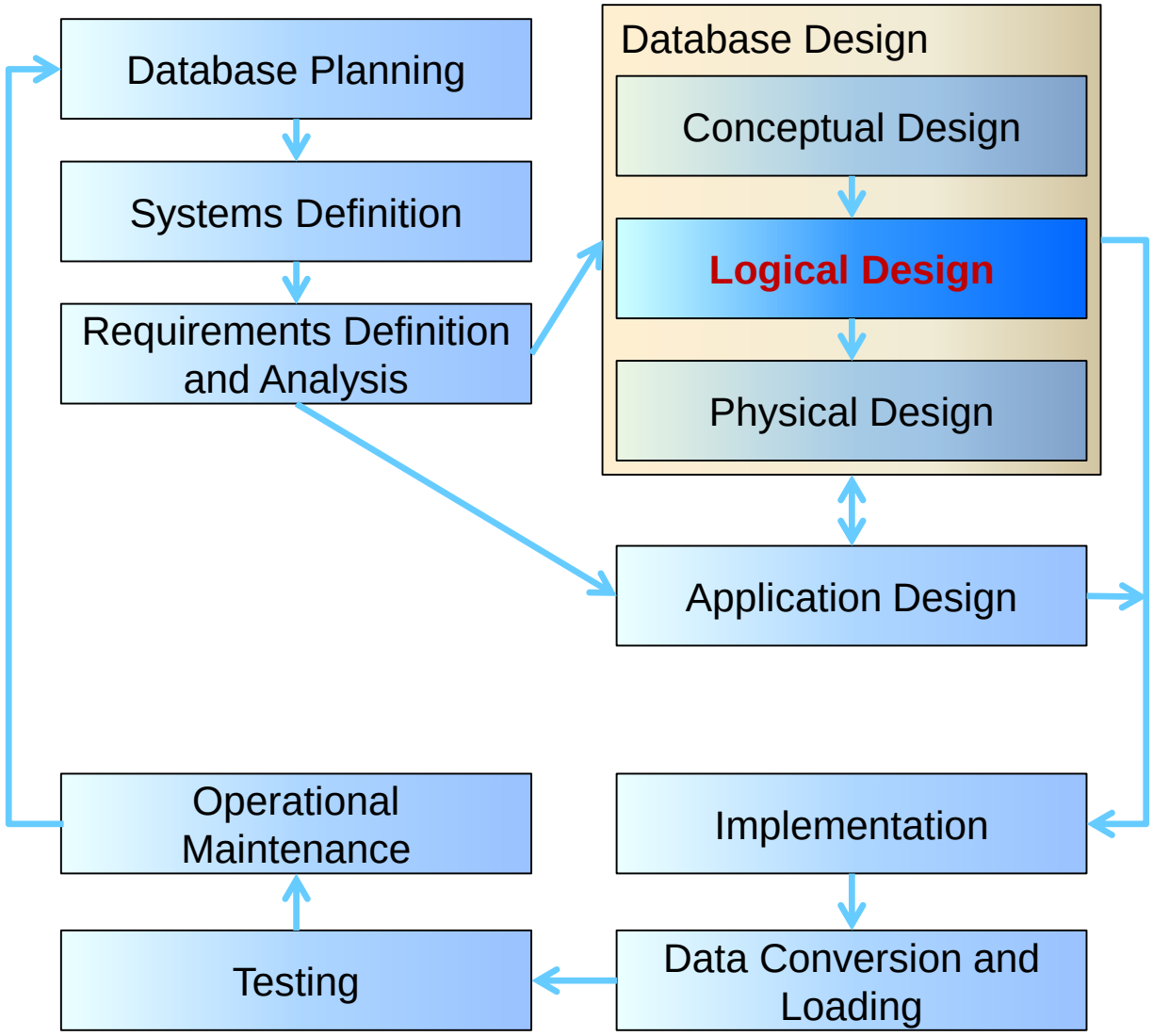
Database Development Lifecycle



- High-level, first-pass model of entities and their connections
- Typically omits attributes
- Could potentially be implemented in a non-relational database
- Thus can include many-to-many relationships, repeating groups, composite attributes

© The University of Melbourne 2018

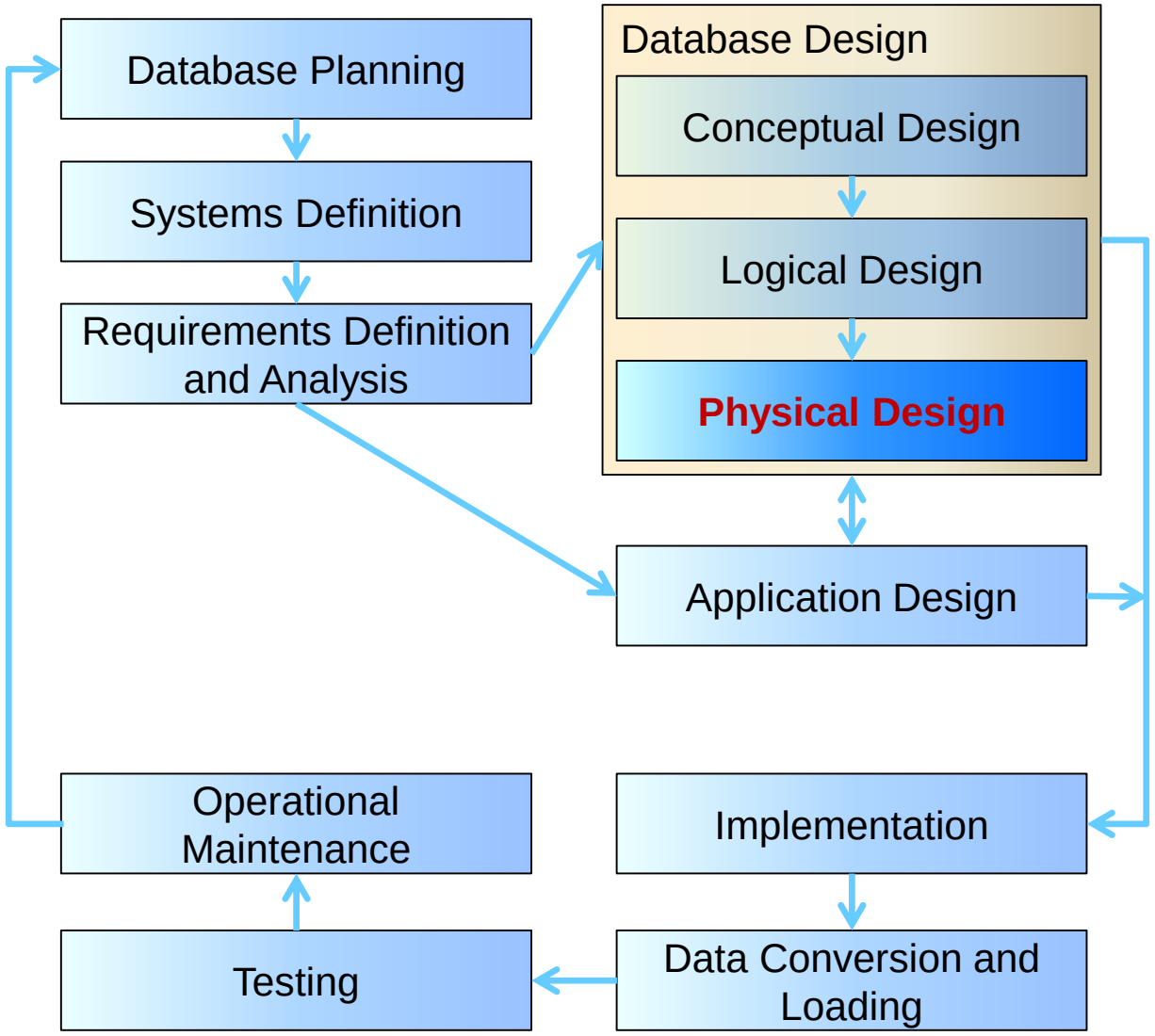
Database Development Lifecycle



- Builds on the conceptual design
- Designing now for a relational database
- Includes columns and keys
- Independent of a specific vendor and other physical considerations

© The University of Melbourne 2018

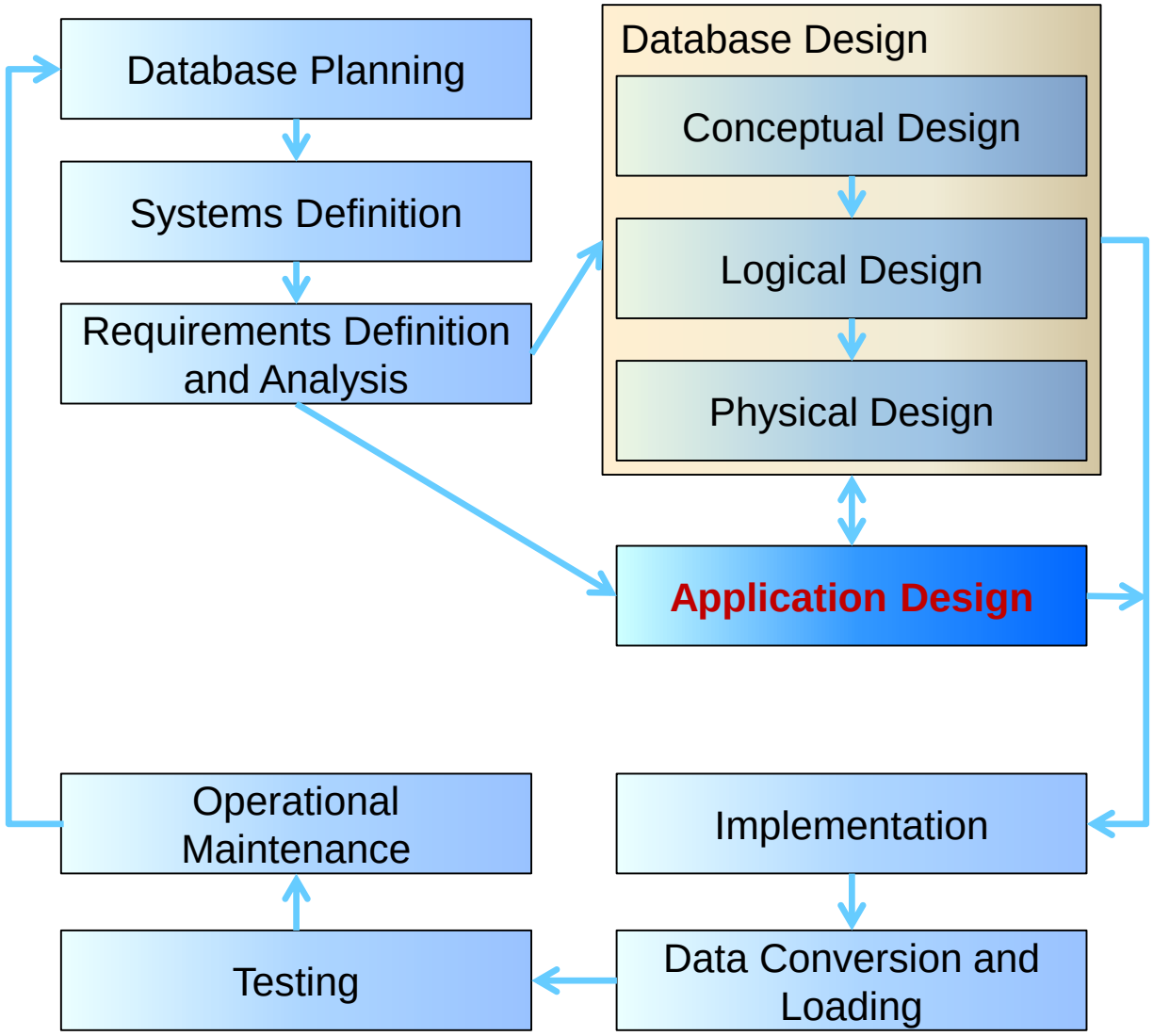
Database Development Lifecycle



- Implements the logical design for a specific DBMS.
- Describes:
  - Base tables
  - Data types
  - Indexes
  - Integrity constraints
  - File organisation
  - Security measures
- We will cover some aspects of physical design

© The University of Melbourne 2018

Database Development Lifecycle

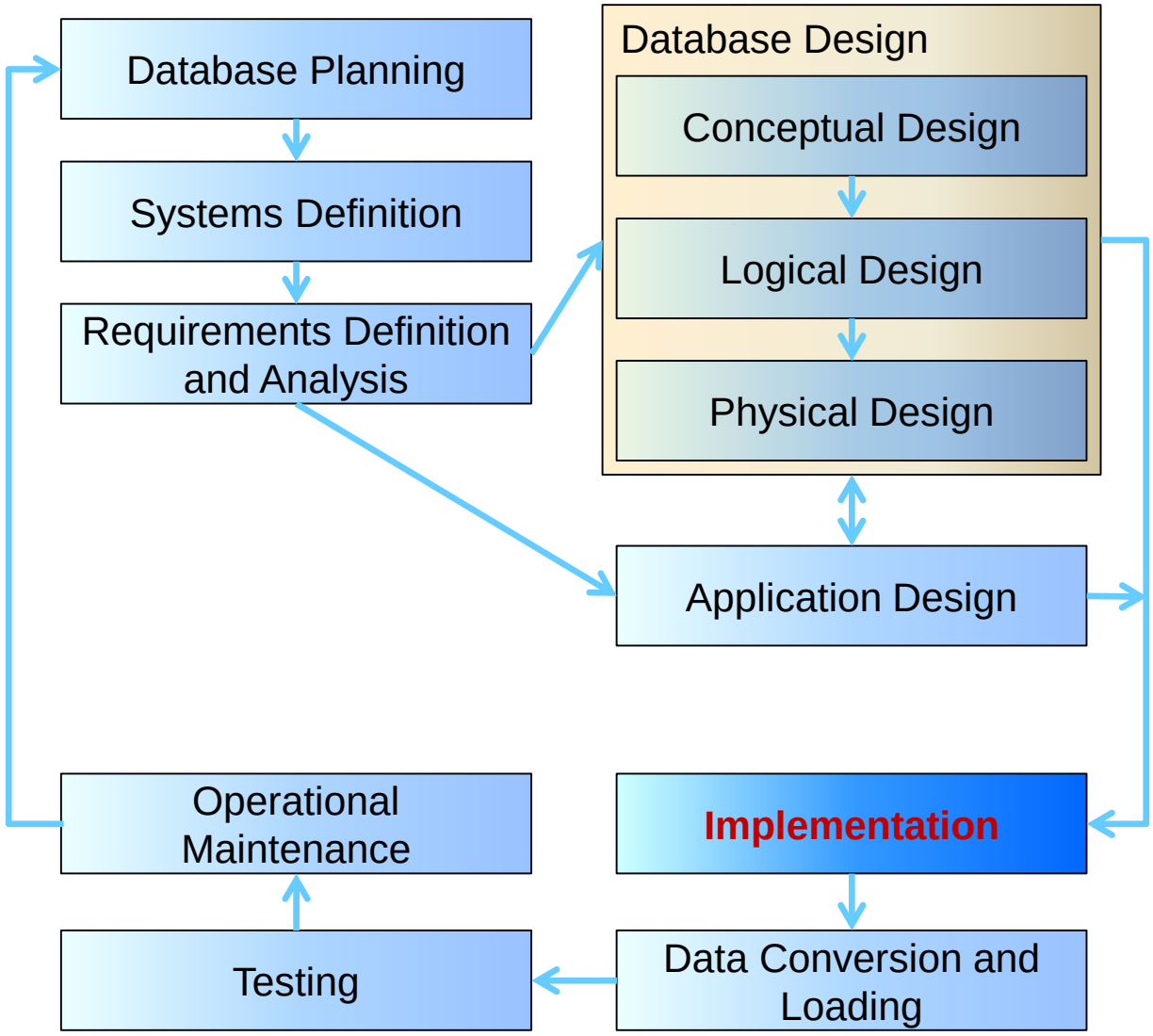


- Done in conjunction with database design
- Design of the interface and application programs that use and process the database

- Mostly outside scope of the course, but discussed in week 7

© The University of Melbourne 2018

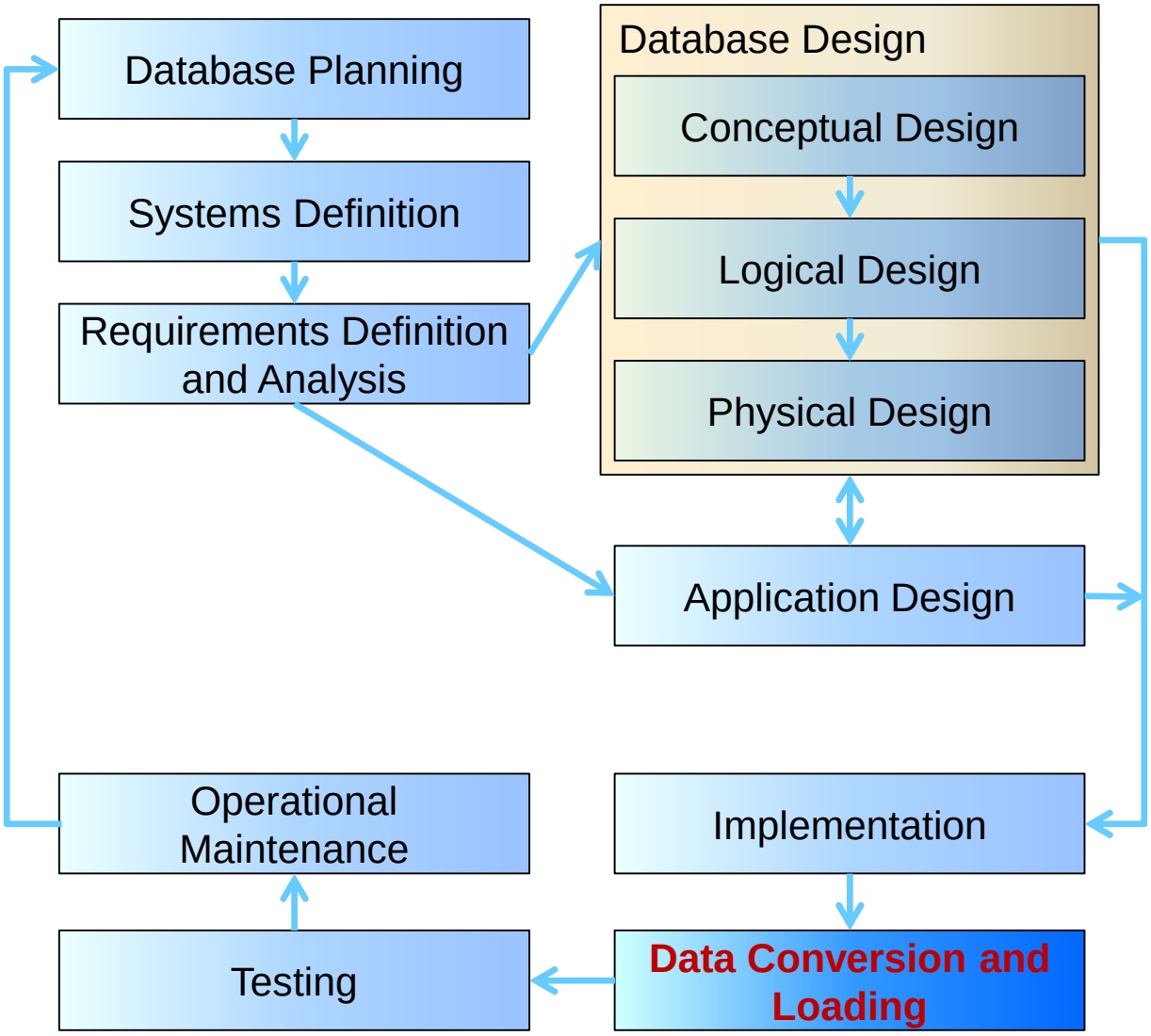
Database Development Lifecycle



- Implementation of the design as a working database

© The University of Melbourne 2018

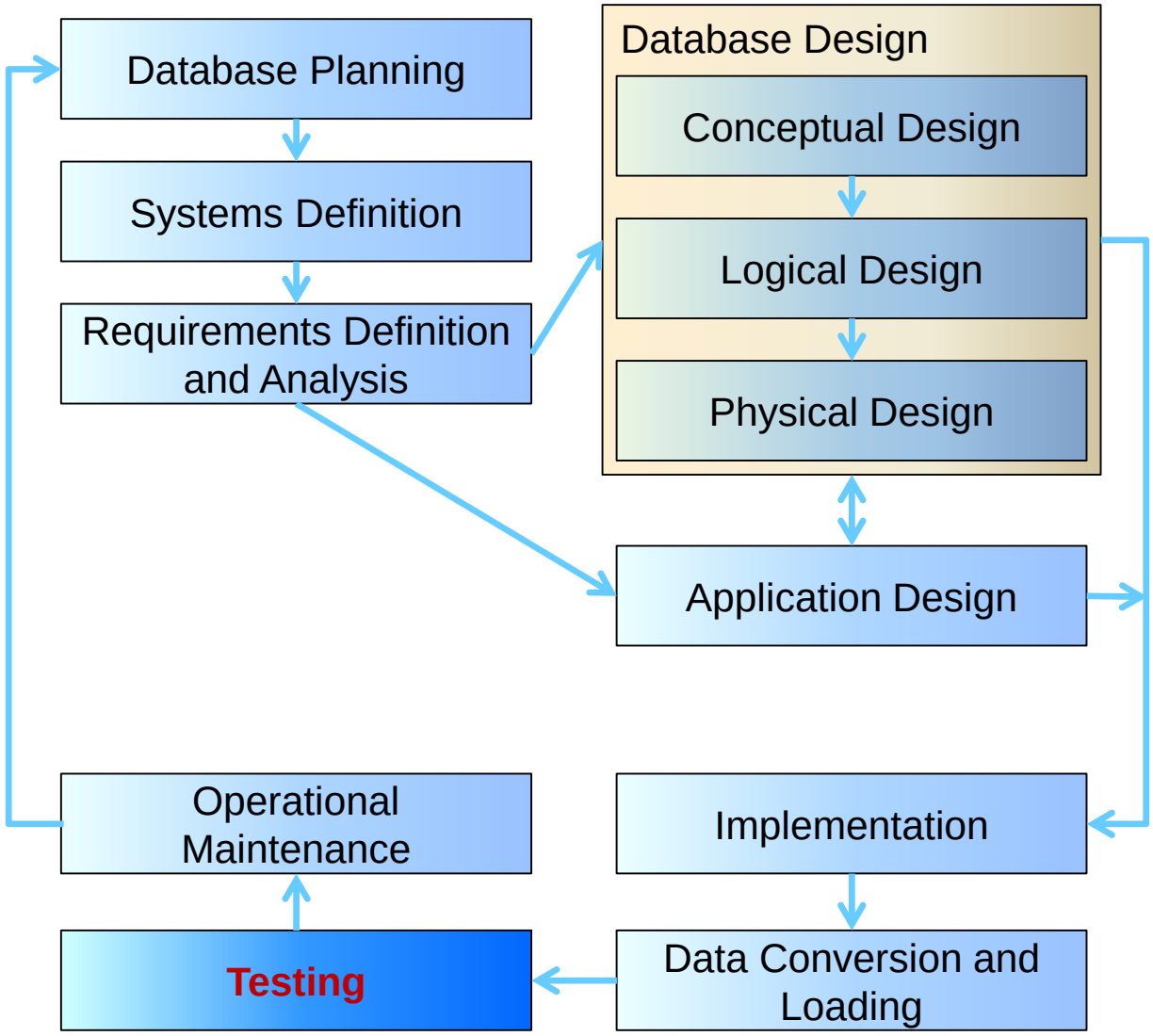
Database Development Lifecycle



- Transfer existing data into the database
- Conversion from old systems
- Non trivial task

© The University of Melbourne 2018

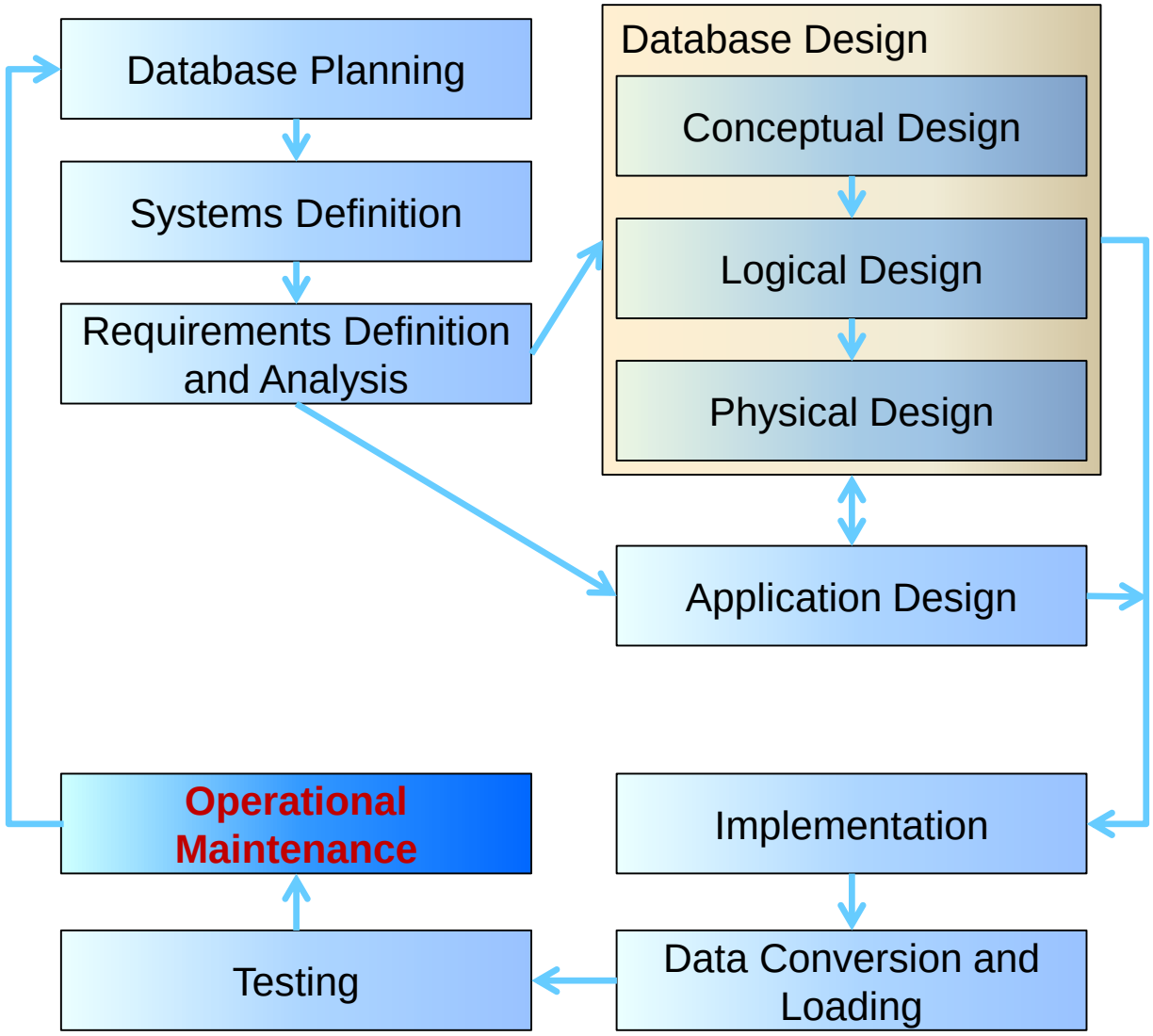
Database Development Lifecycle



- Running the database to find errors in the design / setup
- Other issues also
  - Performance
  - Robustness
  - Recoverability
  - Adaptability
- Mostly outside scope of the course

© The University of Melbourne 2018

Database Development Lifecycle

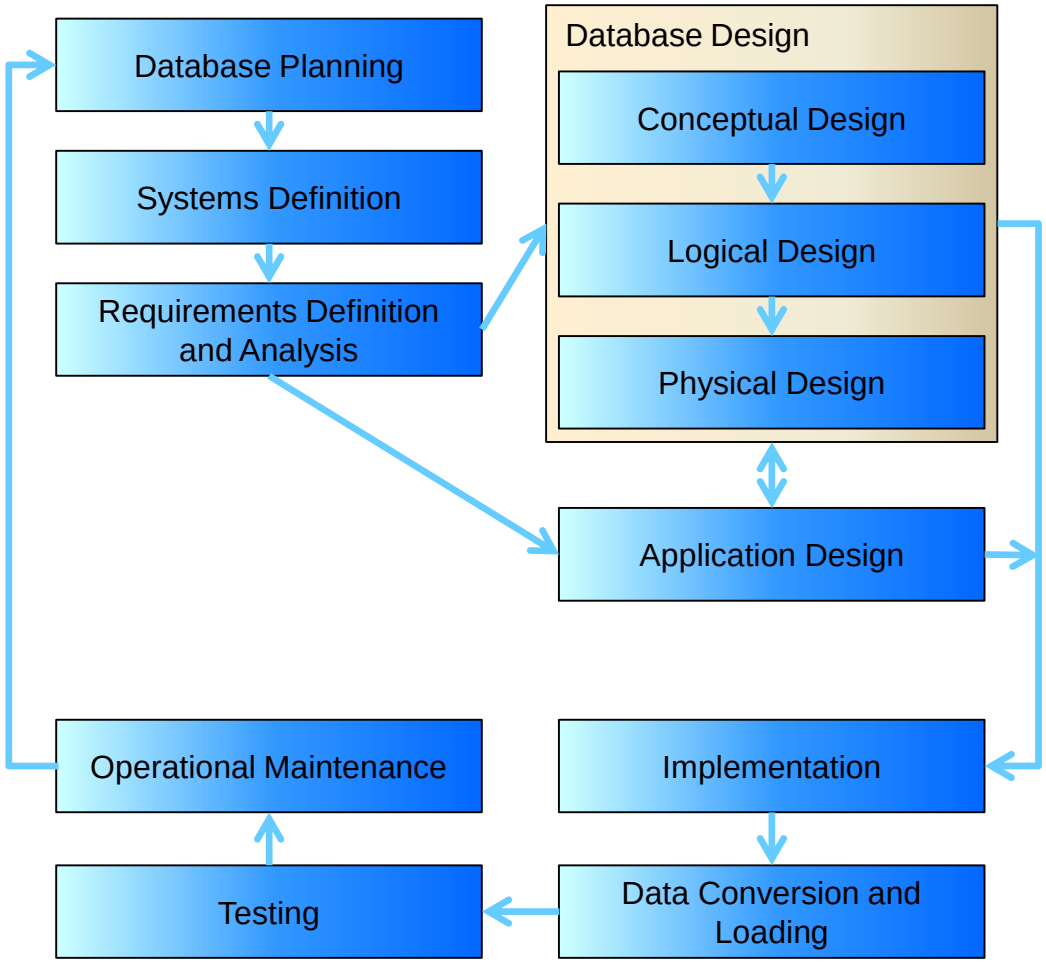


- The process of monitoring and maintaining the database following its commissioning
- Monitoring and improving performance
- Handling changes to requirements

- We will touch on some of these topics later in the semester, especially in week 9

© The University of Melbourne 2018

Summary of database lifecycle



Now we'll work through one example …

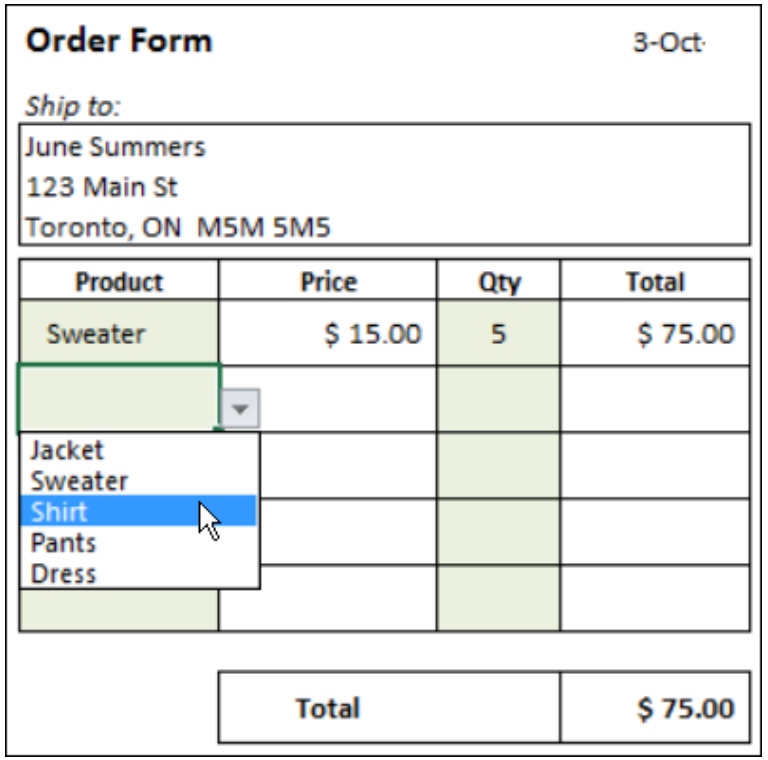
© The University of Melbourne 2018

Case Study: design the db

Data Modelling

Case for this lecture: Orders system

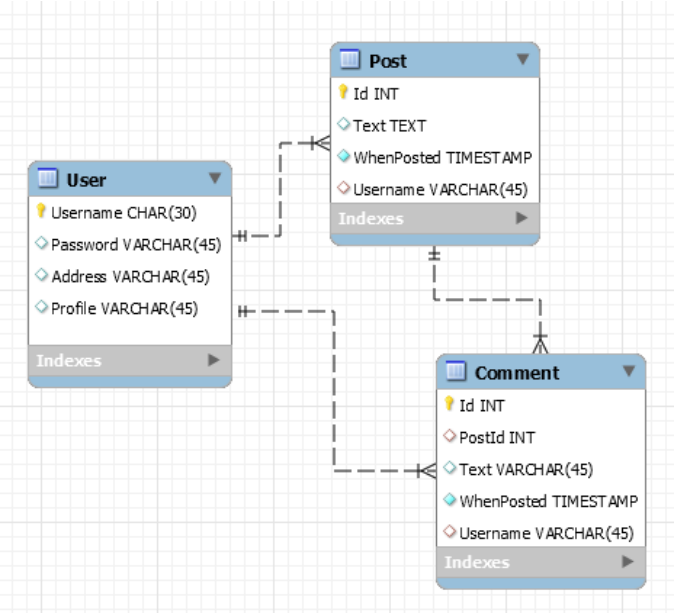
- Our company sells many products. About each product we record its id, name, and price.
- We have many customers. About each customer we record their customer id, name, and address.
- Customers place orders for products. Each order is placed by one customer on a particular date. Over time a customer may place several orders, though some may register but not place any orders.
- Each order must contain at least one order-item, but may contain several. Each order-item records a quantity ordered of one product.



© The University of Melbourne 2018

Recall from week 1: Data Modelling

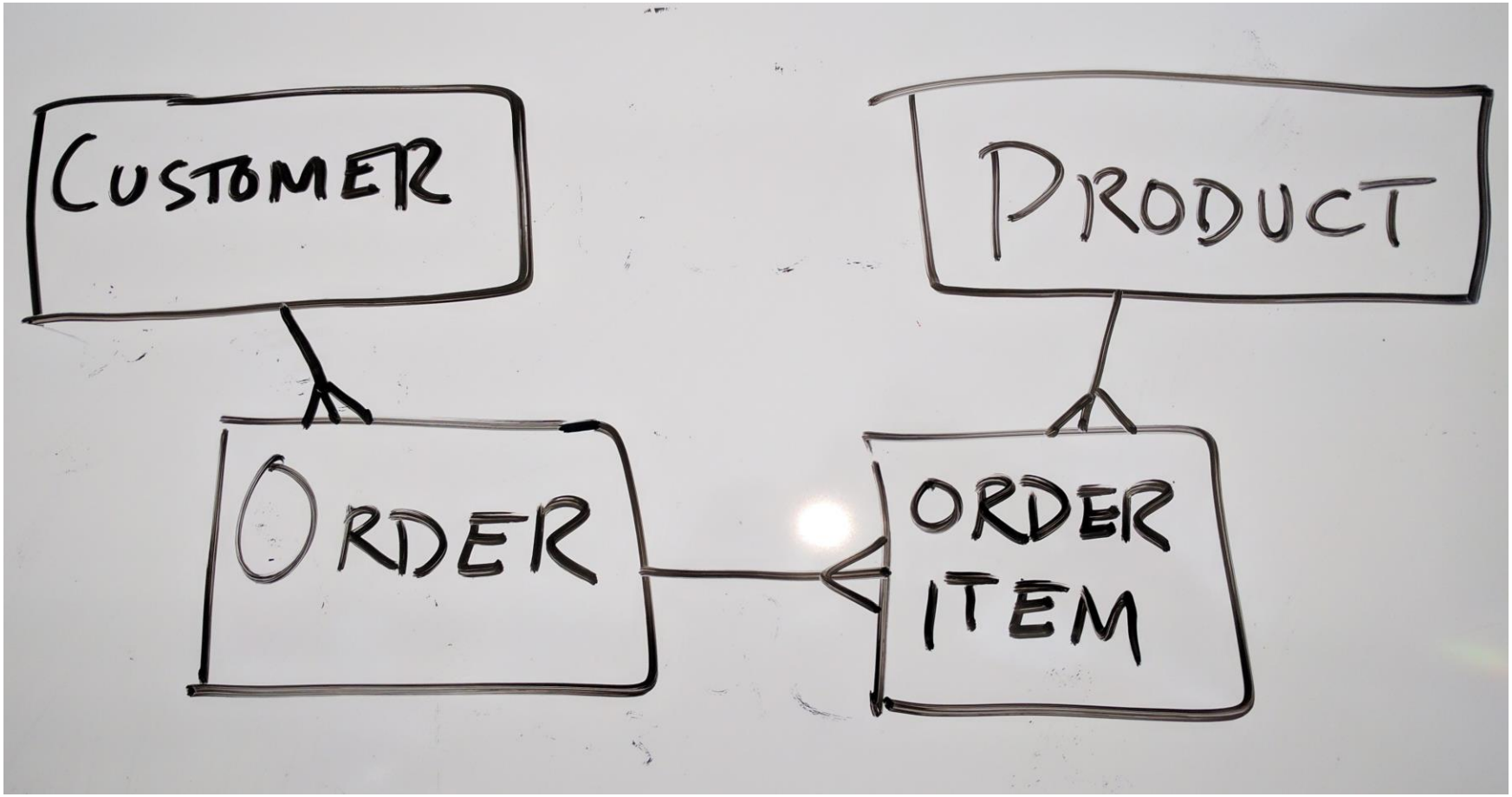
1. What are the entities that need to be tracked?
2. What information will be recorded about each entity?
3. What are the relationships between entities?
4. What are the cardinalities of relationships?



© The University of Melbourne 2018

Conceptual Data Model

(we will create this together during the lecture)

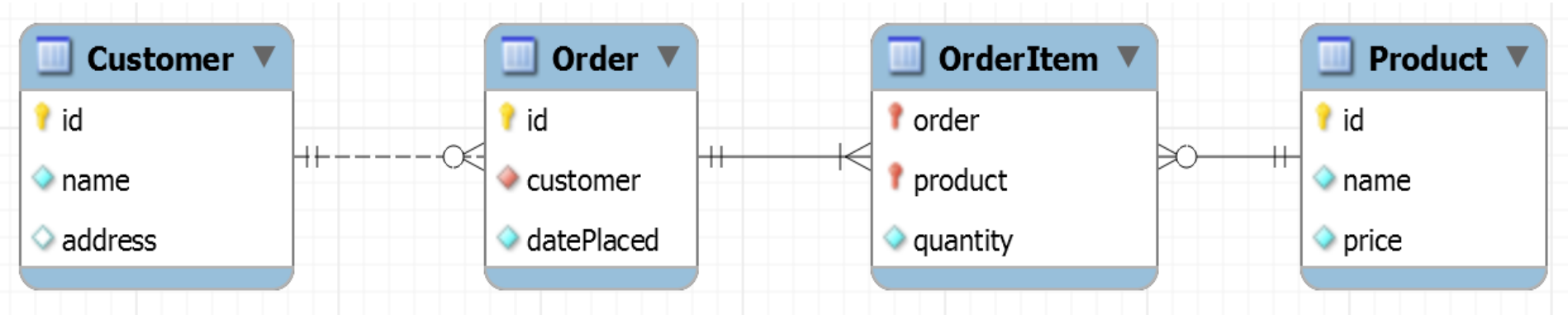


© The University of Melbourne 2018

Logical Data Model

(we will create this together during the lecture)

New question:
What is the *primary key* of each table?

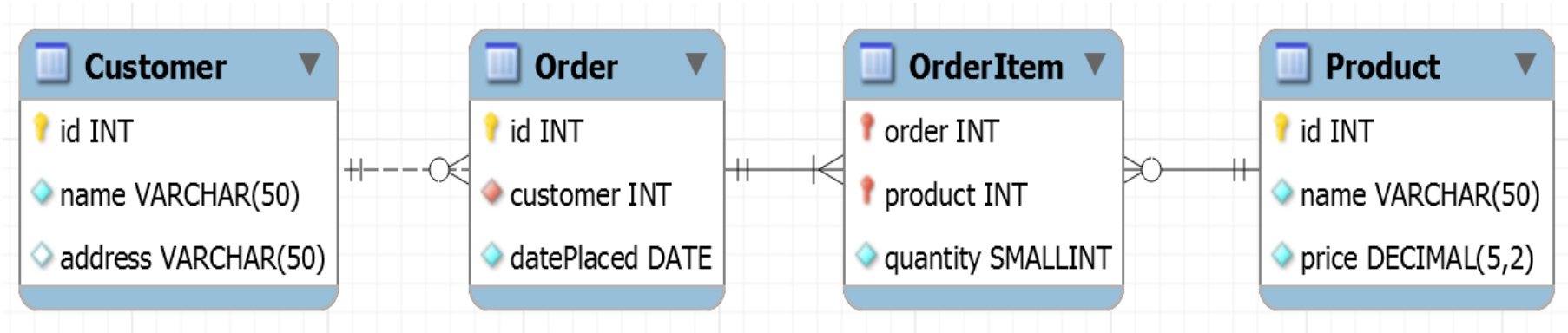


© The University of Melbourne 2018

Physical Model

(we will create this together during the lecture)

New question:
What is the *data type* of each column?



© The University of Melbourne 2018

Case Study: implement the db

SQL

Create tables

Can be done in several ways:

1. manual DDL commands
2. SQL "setup script" (right)
3. forward engineer from MySQL Workbench and similar tools

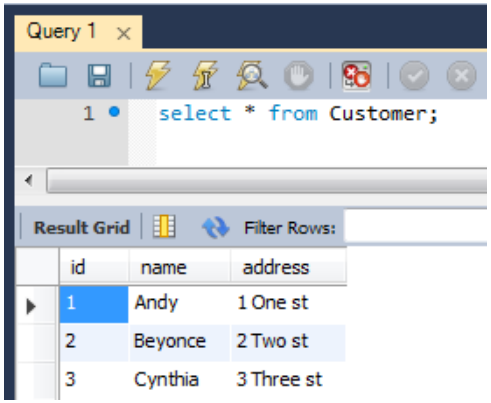
```
CREATE TABLE IF NOT EXISTS `Customer` (
  `id` INT NOT NULL,
  `name` VARCHAR(50) NOT NULL,
  `address` VARCHAR(50) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS `Order` (
  `id` INT NOT NULL,
  `customer` INT NOT NULL,
  `datePlaced` DATE NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Order_Customer_idx` (`customer` ASC),
  CONSTRAINT `fk_Order_Customer`
    FOREIGN KEY (`customer`)
    REFERENCES `Customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS `Product` (
  `id` INT NOT NULL,
  `name` VARCHAR(50) NOT NULL,
  `price` DECIMAL(5,2) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS `OrderItem` (
  `order` INT NOT NULL,
  `product` INT NOT NULL,
  `quantity` SMALLINT NOT NULL,
  PRIMARY KEY (`order`, `product`),
  INDEX `fk_OrderItem_Product1_idx` (`product` ASC),
  CONSTRAINT `fk_OrderItem_Order1`
    FOREIGN KEY (`order`)
    REFERENCES `Order` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_OrderItem_Product1`
    FOREIGN KEY (`product`)
    REFERENCES `Product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```


© The University of Melbourne 2018

Insert data

Can be done in several ways:

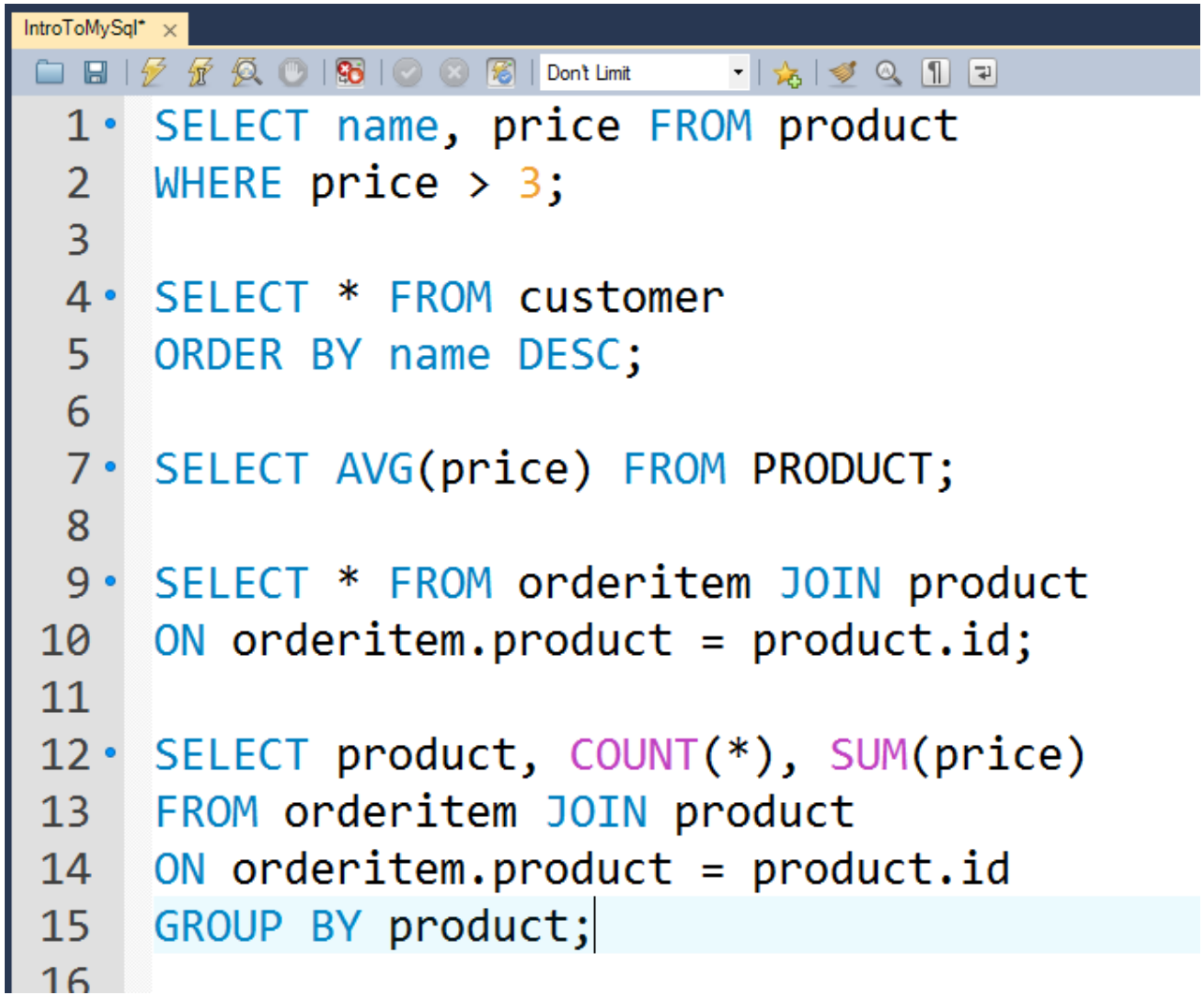
1. manual Insert commands
2. SQL script file (right)
3. Insert in Workbench model
4. via application software




```
/* INSERT DATA */

insert into Customer values (1,'Andy','1 One st');
insert into Customer values (2,'Beyonce','2 Two st');
insert into Customer values (3,'Cynthia','3 Three st');

insert into Product values (1,'apple',1.11);
insert into Product values (2,'bread',2.22);
insert into Product values (3,'cabbage',3.33);
insert into Product values (4,'dates',4.44);

insert into `Order` values (1,1,'2017-6-1');
insert into `Order` values (2,2,'2017-6-2');
insert into `Order` values (3,3,'2017-6-3');
insert into `Order` values (4,1,'2017-6-4');
insert into `Order` values (5,2,'2017-6-5');

insert into OrderItem values (1,1,1);
insert into OrderItem values (1,2,2);
insert into OrderItem values (1,3,3);
insert into OrderItem values (2,2,4);
insert into OrderItem values (2,4,5);
insert into OrderItem values (3,1,6);
insert into OrderItem values (3,2,7);
insert into OrderItem values (3,3,8);
insert into OrderItem values (3,4,9);
insert into OrderItem values (4,4,10);
insert into OrderItem values (5,3,9);
```


© The University of Melbourne 2018

Query the data



© The University of Melbourne 2018

Change the data

demoQueries*

Don't Limit

```
1  UPDATE product
2  SET Price = 3.50
3  WHERE name = 'cabbage'; /* change price of cabbage */
4
5  INSERT INTO `order`
6  values (6, 1, '2017-7-13');  /* add a new order */
7
8  INSERT INTO orderitem
9  values (6, 2, 20), (6, 3, 30); /* add items to our new order */
10
11 DELETE from product
12 WHERE name = 'dates';   /* delete one product */
13
```


© The University of Melbourne 2018

Next 3 weeks

- More detailed understanding of database design
  - Conceptual design
  - Logical design
  - Physical design
- More detailed understanding of SQL
  - Operations on a single table
  - Joining multiple tables

© The University of Melbourne 2018