INFO20003 Database Systems

Dr Renata Borovica-Gajic

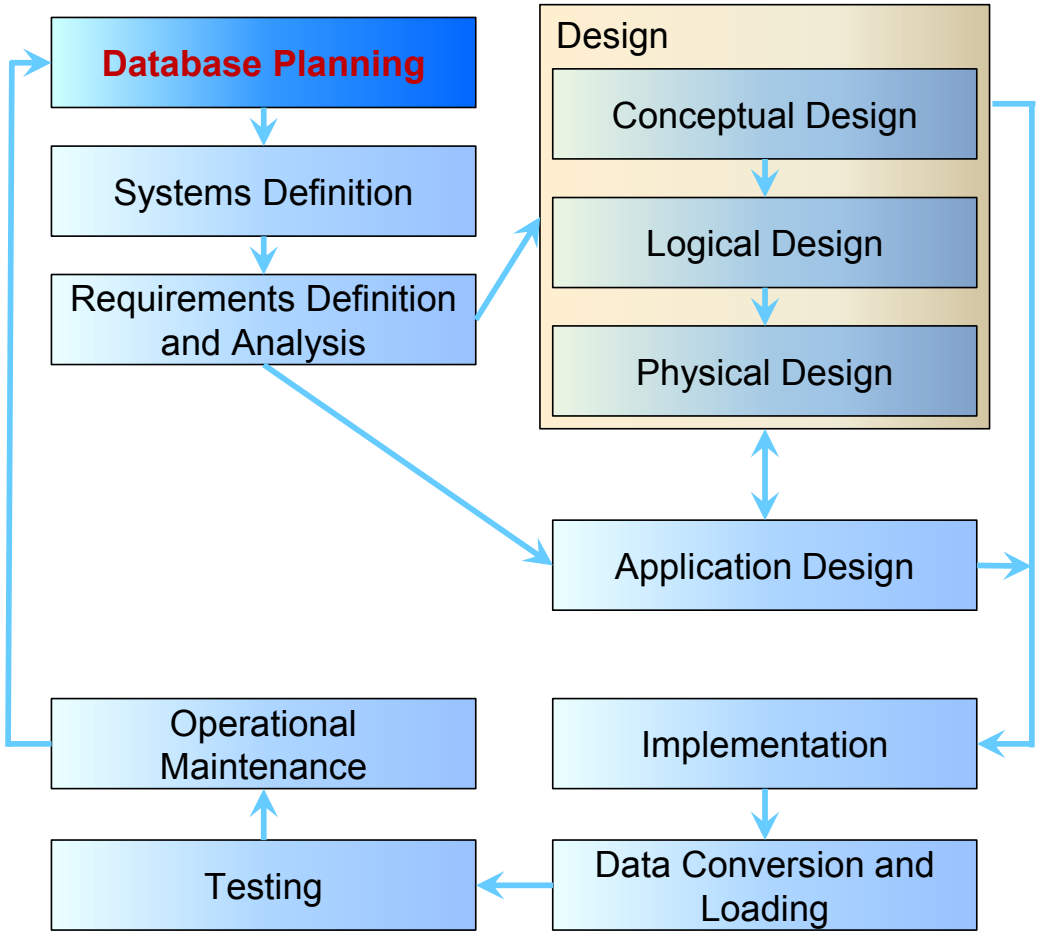
Lecture 02
Database Development Process

Week 1

Coverage

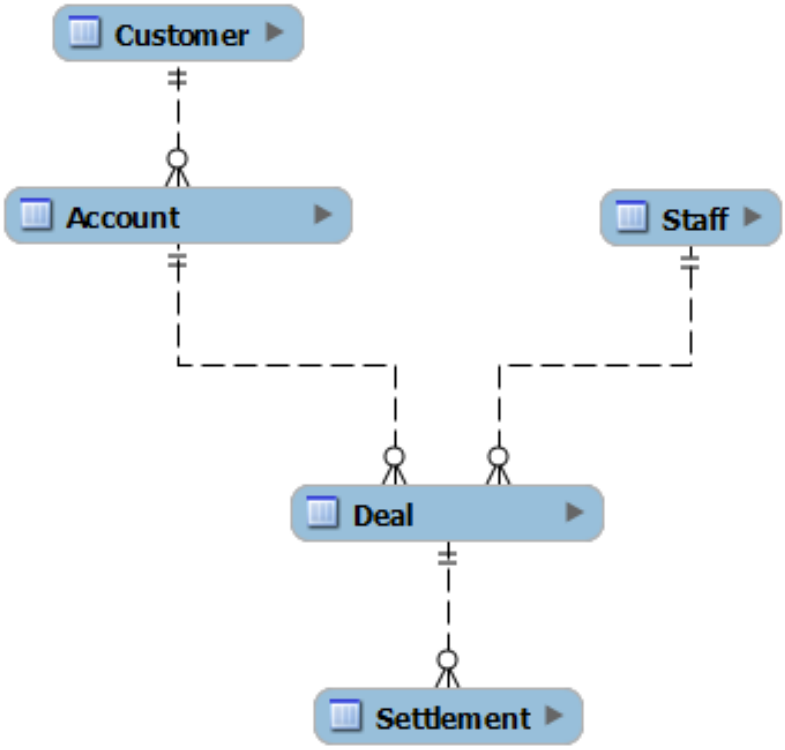
- How database applications are developed
  - The development lifecycle
  - Focus on database *design*
    - Conceptual design
    - Logical design
    - Physical design

Database Development Lifecycle



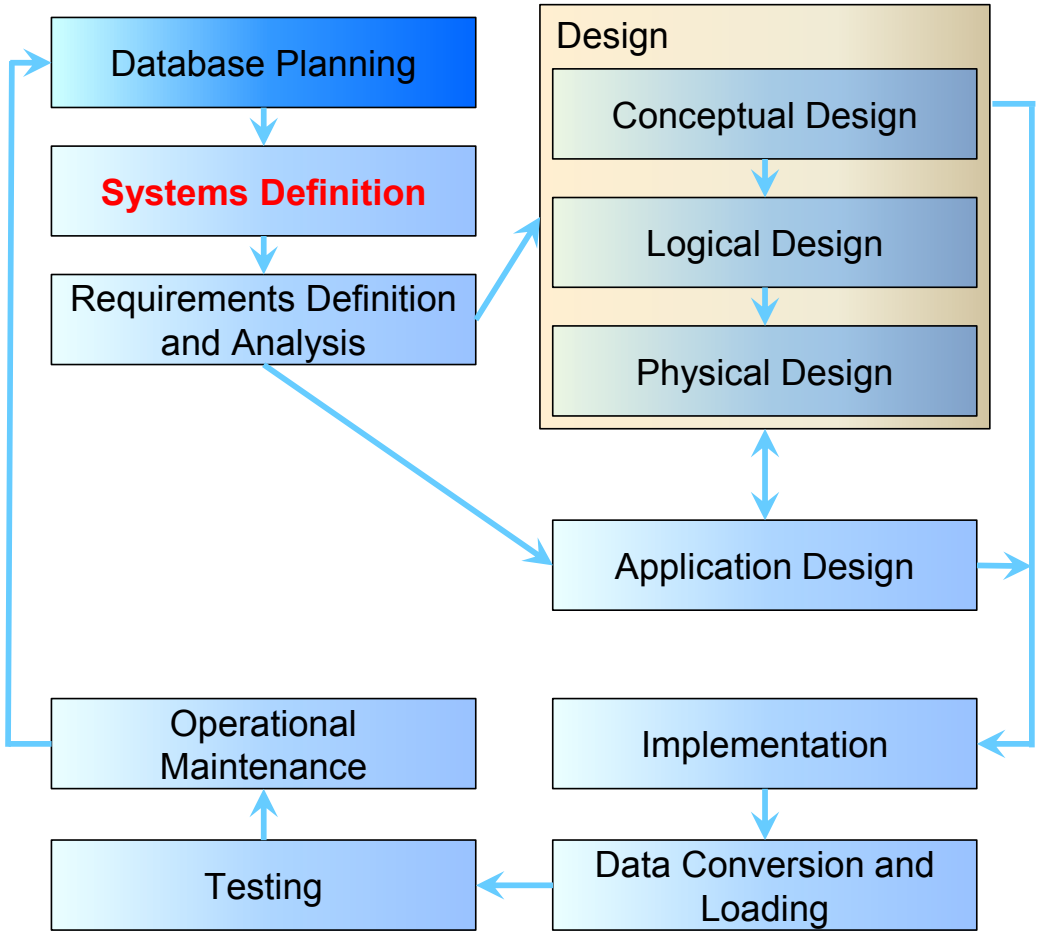
- Planning how to do the project.
  - How does the enterprise work
  - Enterprise data model
- Outside scope of the course

Example Enterprise Data Model – Investment Banking



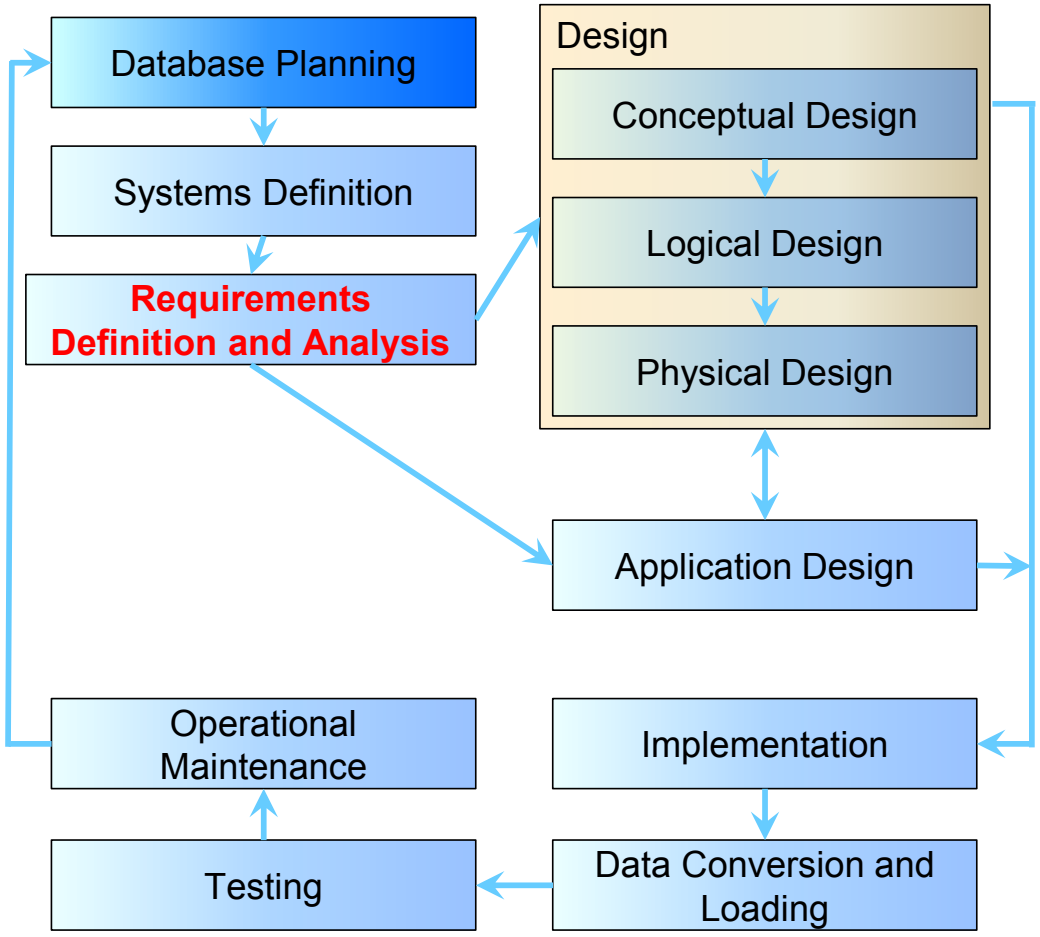
- A top level perspective on the data requirements
- Each box (subject area) would have a data model

Database Development Lifecycle



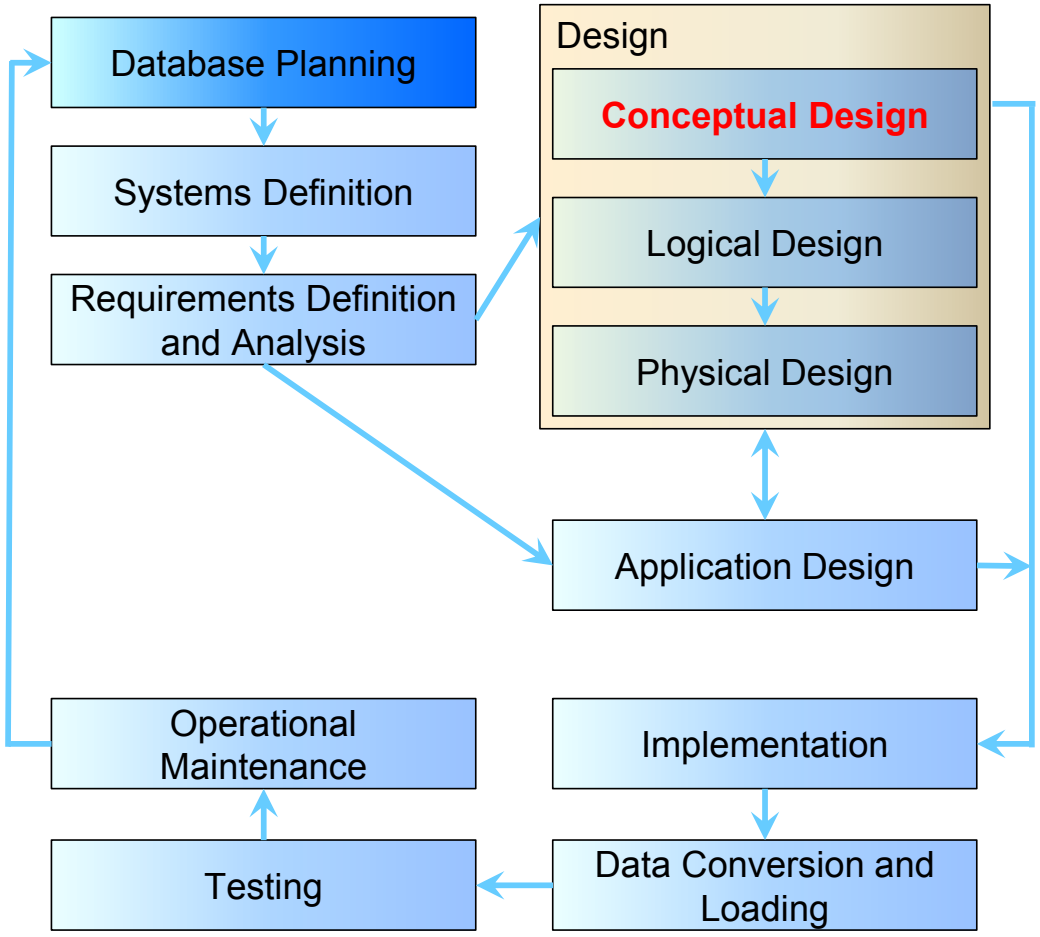
- Specifying scope and boundaries
  - Users
  - Application areas
- How does the system interfere with other organisational systems
- Outside scope of the course (slightly)

Database Development Lifecycle



- Collection and analysis of requirements for the new system
- You will be given the requirements, but you will need to understand these!
- You may need to ask requirement questions about what you are given (for the assignment you state your assumptions)

Database Development Lifecycle



- Construction of a model of the data used in the database – independent of all physical considerations
- Data Models
  - ER Diagrams

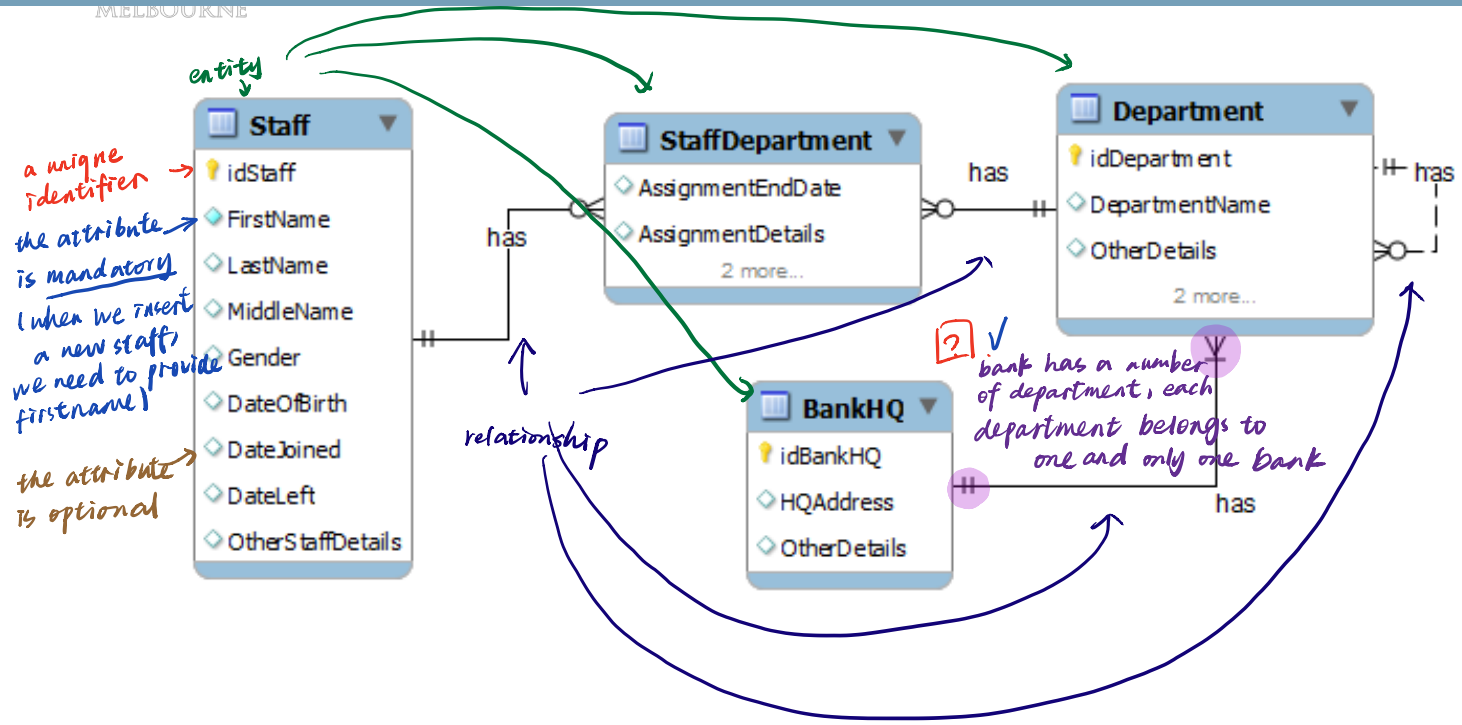
Analysis of the problem

Business rule

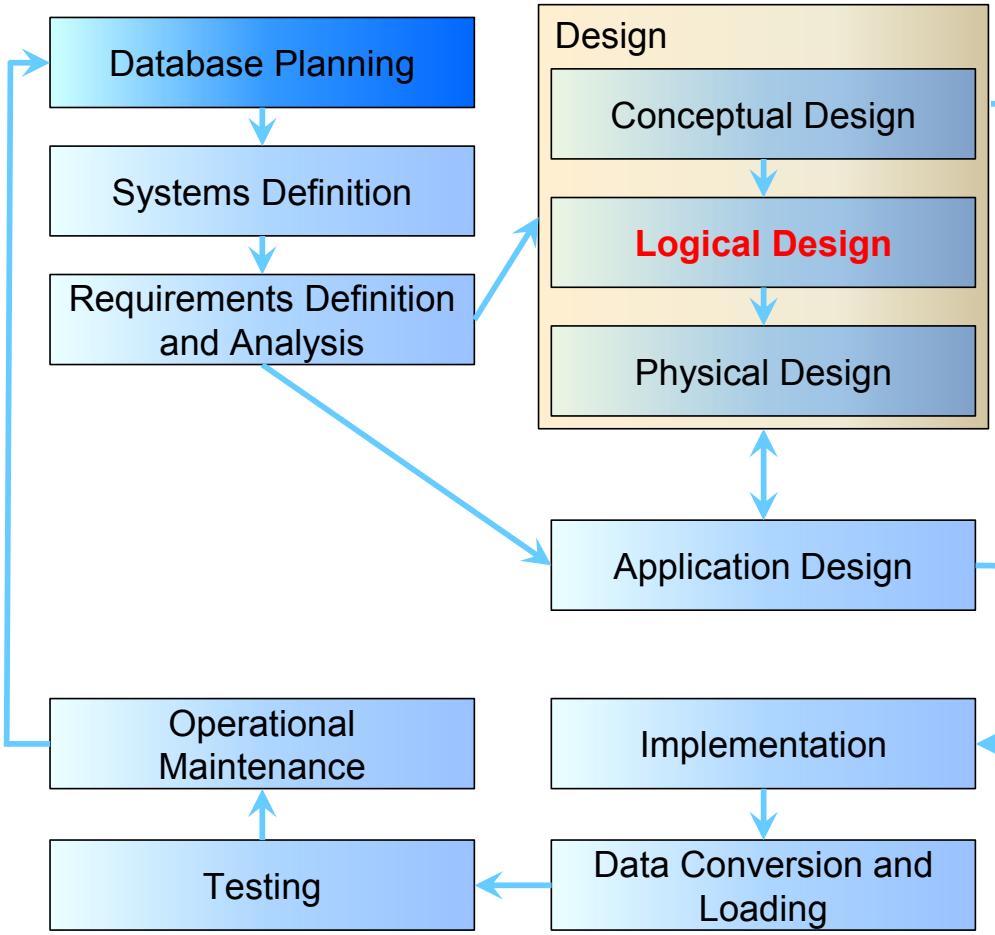
- An investment bank has *a number* of branches. Within *each branch a number of departments* operate and are *structured in a hierarchical manner*. The bank employs around 3000 staff *who are assigned to work in the* various departments across the branches.

- We need a database to record staff details including which department and branch they are assigned...

Example Conceptual Data Model (ER) – Investment Banking



Database Development Lifecycle



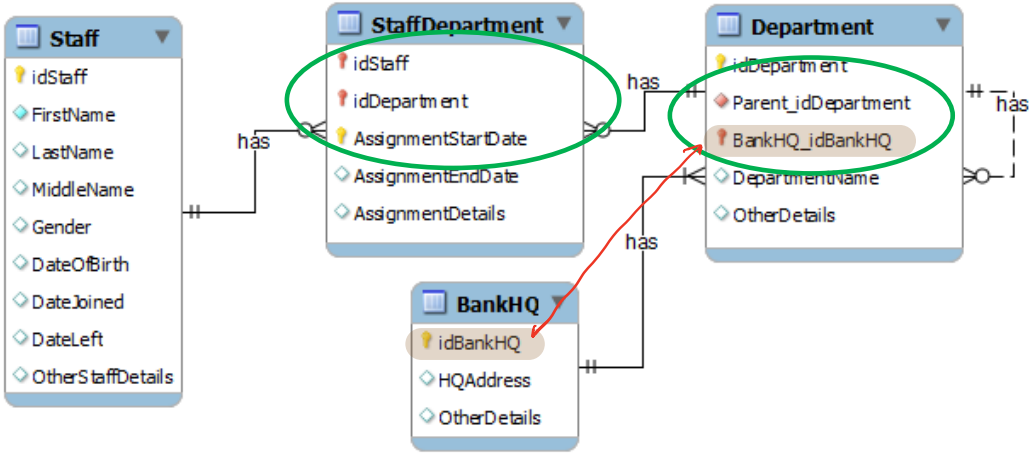
- Construction of a (relational) model of the data based on the conceptual design
- Independent of a specific database and other physical considerations

(we haven't decide which database to use, but we have decide that model is relational

store data in json ? xml ?
relational model ?
↓
(a set of table)

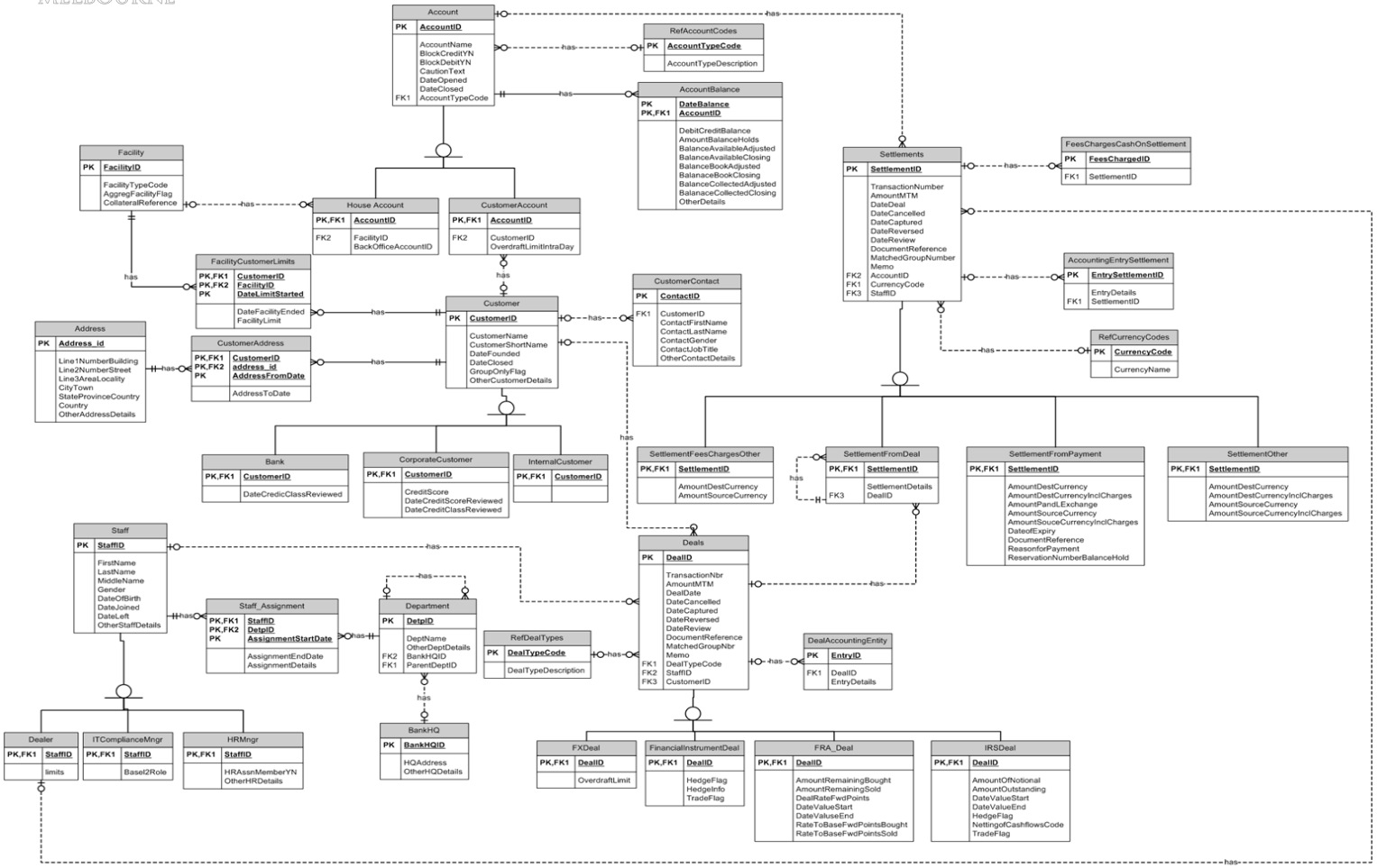
can choose non-relational database ?

Example Logical Data Model – Investment Banking

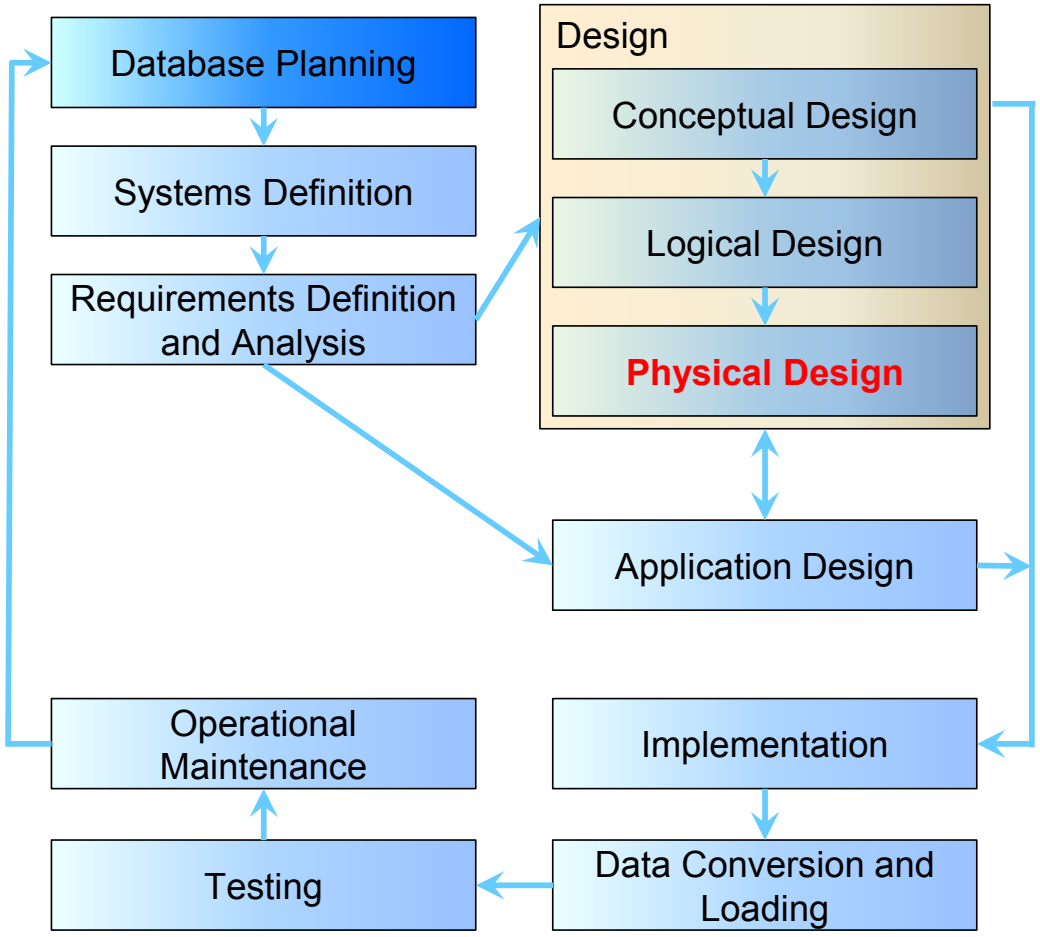


Changes from Conceptual Model (ER)

Example Logical Data Model – Investment Banking (Complete)

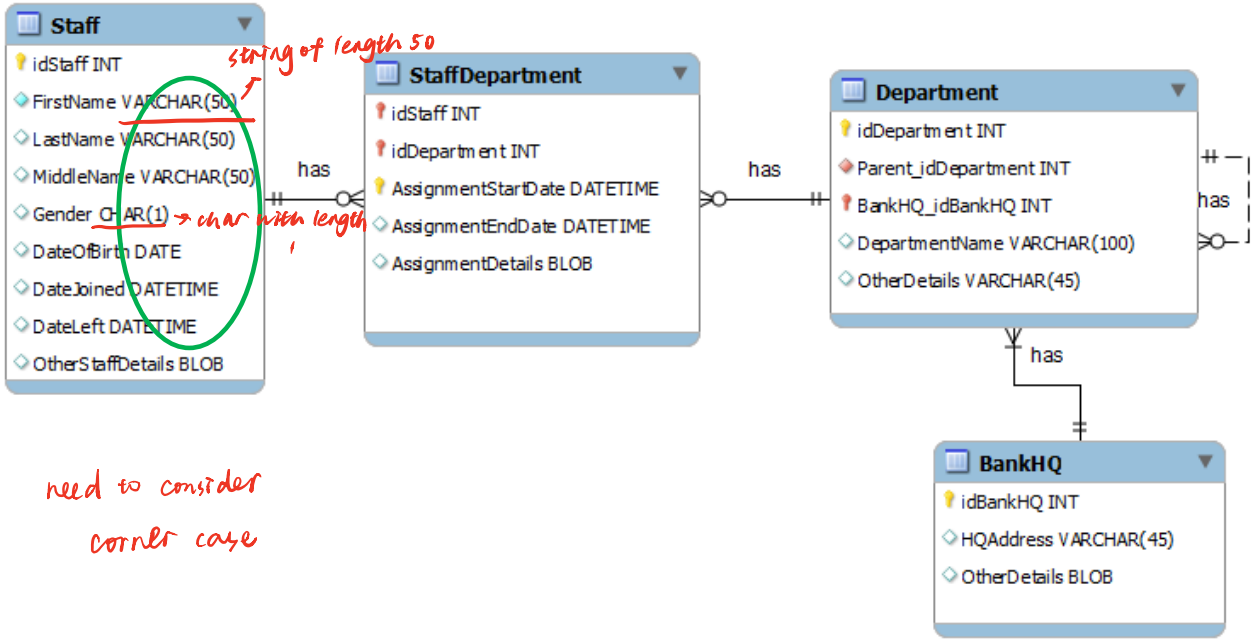


Database Development Lifecycle



- A description of the implementation of the logical design – for a *specific* DBMS.
- Describes: (which database system we use to build model)
  - Basic relations (data types)
  - File organisation
  - Indexes
- Although we don't do full physical design, we discuss some of the issues (e.g. data types)

Example Physical Model – Investment Banking (Staff)



Choosing Data Types

- Types help the DBMS store and use information efficiently
  - Can make assumptions in computation
  - Consistency is guaranteed (by specify data type).
- Minimise storage space
- Need to consider
  - Can you store all possible values
  - Can the type you choose support the data manipulation required
- Selection of types may improve data integrity

street number → string or VARCHAR

postcode → string or VARCHAR

eg: 25B
① no meaning to summerize
② in some country, postcode can contain letters

① corner case
② what sort of manipulation we will do with data

Example of Data Dictionary

- We do the data dictionary as an ongoing process during analysis and design of the database
- Example of what is required

Key identifier or not	**Attribute**	**Data Type**	**Not Null** mandatory ?	**Unique**	**Description**
Type of key Is it a primary key or a foreign key (leave blank if neither)	Name of Attribute	Data type of attribute	If the field is required or is optional	Must the value in the field be unique for that field	A description of the attribute giving any information that could be useful to the database designers or to the application developers. This would include things like attribute sizes, valid values for an attribute, information about coding for this attribute etc.

Example of Partial Data Dictionary

Key	Attribute	Data Type	Not Null	Unique	Description
PK	StaffID	Integer	Y	Y	ID number of the staff member, should be 5 in length. This is the primary identifier (key) of the table.
	FirstName	VarChar			The first given name of the staff member, up to 100 characters.
	LastName	VarChar	Y		The family name of the staff member, up to 100 characters. This must exist for every staff member
	Gender	ENUM	Y		The gender of the staff member, valid values are only “Male” or “Female” (???). An enumerated data type should be used if possible. This should be limited in applications using this field also.
	DateOfBirth	DateTime	Y		This is when the staff member was born. Needs dd/mm/yyyy format.
	...				

MYSQL Data Types (some)

- Character Types
  - CHAR(M): A fixed-length string, right-padded with spaces. The range of M is 0 to 255.
  - VARCHAR(M): A variable-length string. The range of M is 1 to 65535. (its 255 max. in MySQL 4).
  - BIT, BOOL, CHAR: CHAR(1).
  - BLOB, TEXT: up to 65535 bytes (for blob) or characters (for text).
  - ENUM ('value1','value',…) up to 65,535 members.
  - SET ('value1','value2', …) up to 64 members.
- Integer Types
  - TINYINT[(M)]: Signed (-128 to 127) Unsigned(0 to 255)
  - SMALLINT[(M)]: Signed (-32768 to 32767) Unsigned (0 to 65535)
  - MEDIUMINT[(M)]: Signed (-8388608 to 8388607) Unsigned (0 to 16777215)
  - INT[(M)] / INTEGER[(M)]: Signed (-2147483648 to 2147483647) Unsigned (0 to 4294967295)
  - BIGINT[(M)]:Signed(-9223372036854775808 to 9223372036854775807) Unsigned(0 to 18,446,744,073,709,551,615)

MYSQL Data Types (some)

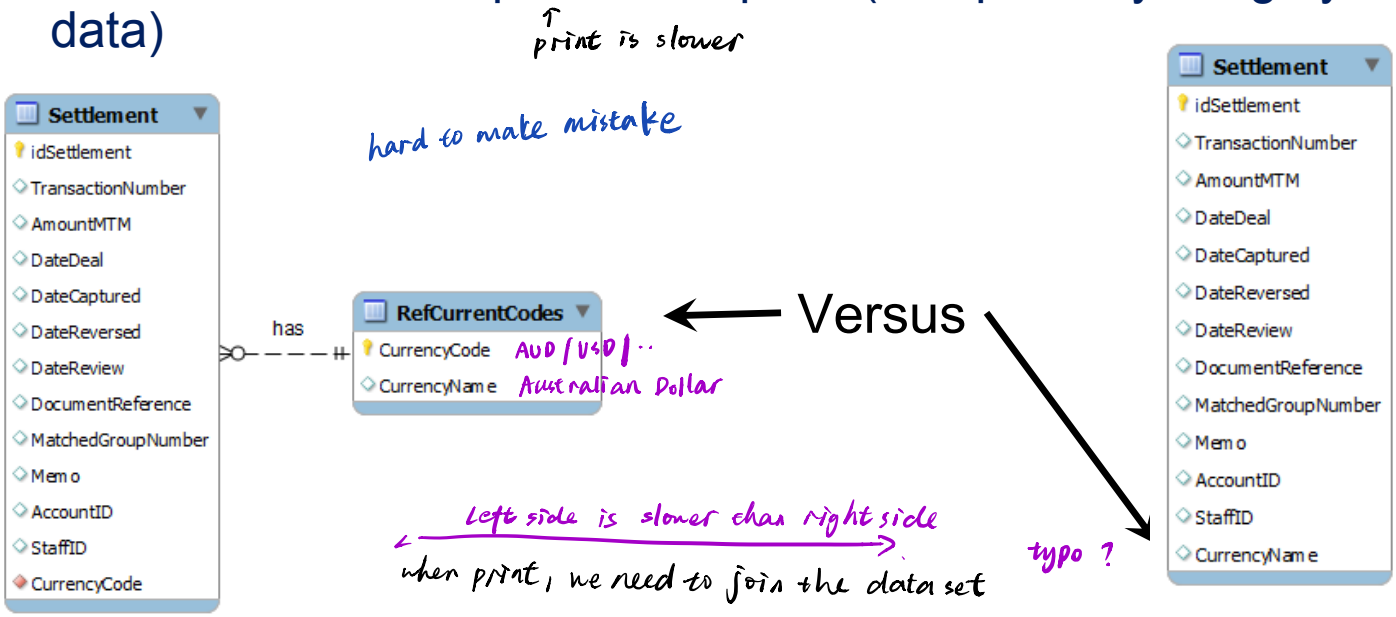
- Real Types
  - FLOAT[(M,D)]: single-precision, allowable values: -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38. M = display width, D = number of decimals.
  - DOUBLE[(M,D)] / REAL[(M,D)]: double-precision, allowable values: -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308.
  - DECIMAL[(M[,D])]: fixed-point type. An unpacked floating-point number. Stored as string. Good for MONEY!

DECIMAL(5,2)
规定存储的值不超5位数字，且小数点后有2位数字。

- Time and Date Types
  - DATE 1000-01-01 to 9999-12-31
  - TIME -838:59:59 to 838:59:59
  - DATETIME 1000-01-01 00:00:00 to 9999-12-31 23:59:59
  - TIMESTAMP 1970-01-01 00:00:00 - ~ 2037 Stored in UTC, converted to local
  - YEAR[4] 1901 to 2155 - A useful function in MySQL: NOW();

Other Physical Design Decisions

- How to store "Look Up"
  - Trade off between speed and space (and possibly integrity of data)

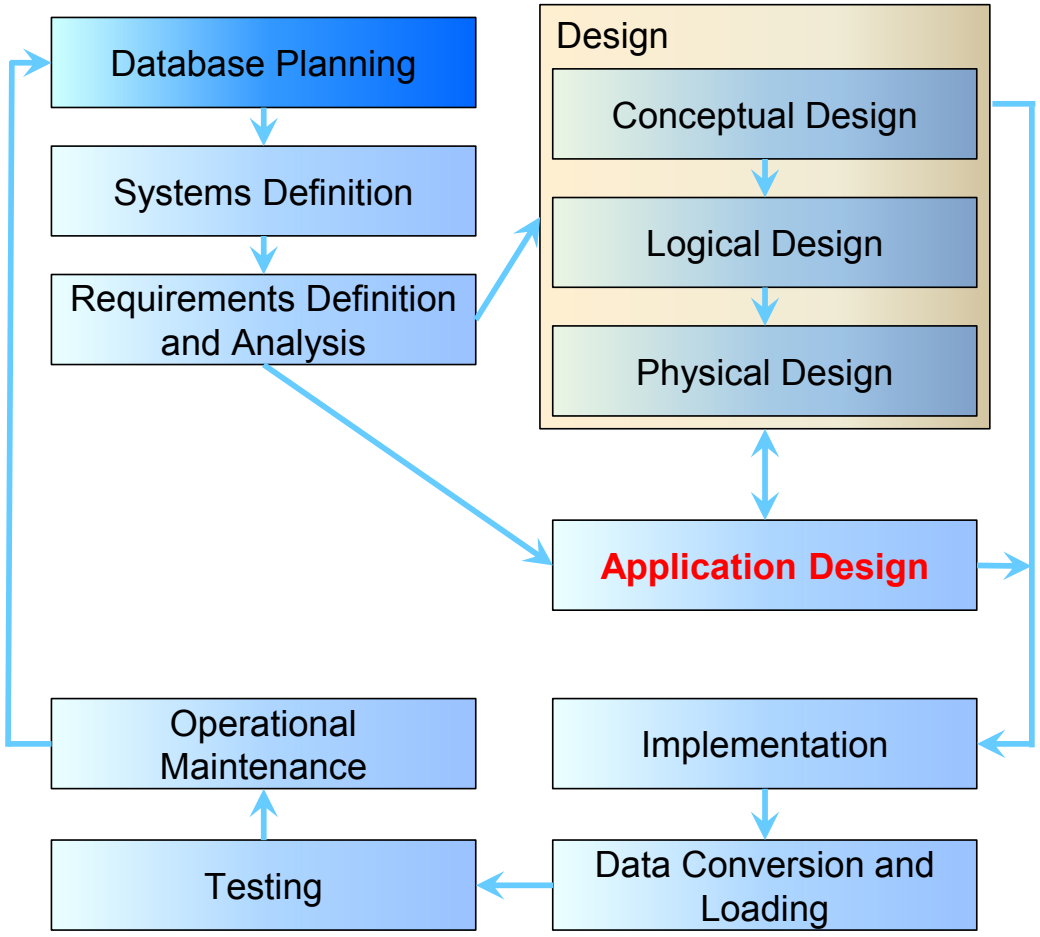


- Data field integrity (ensure fields only contain correct data)
- Handling missing data (concept of NULL data)

Other Physical Design Decisions (2)

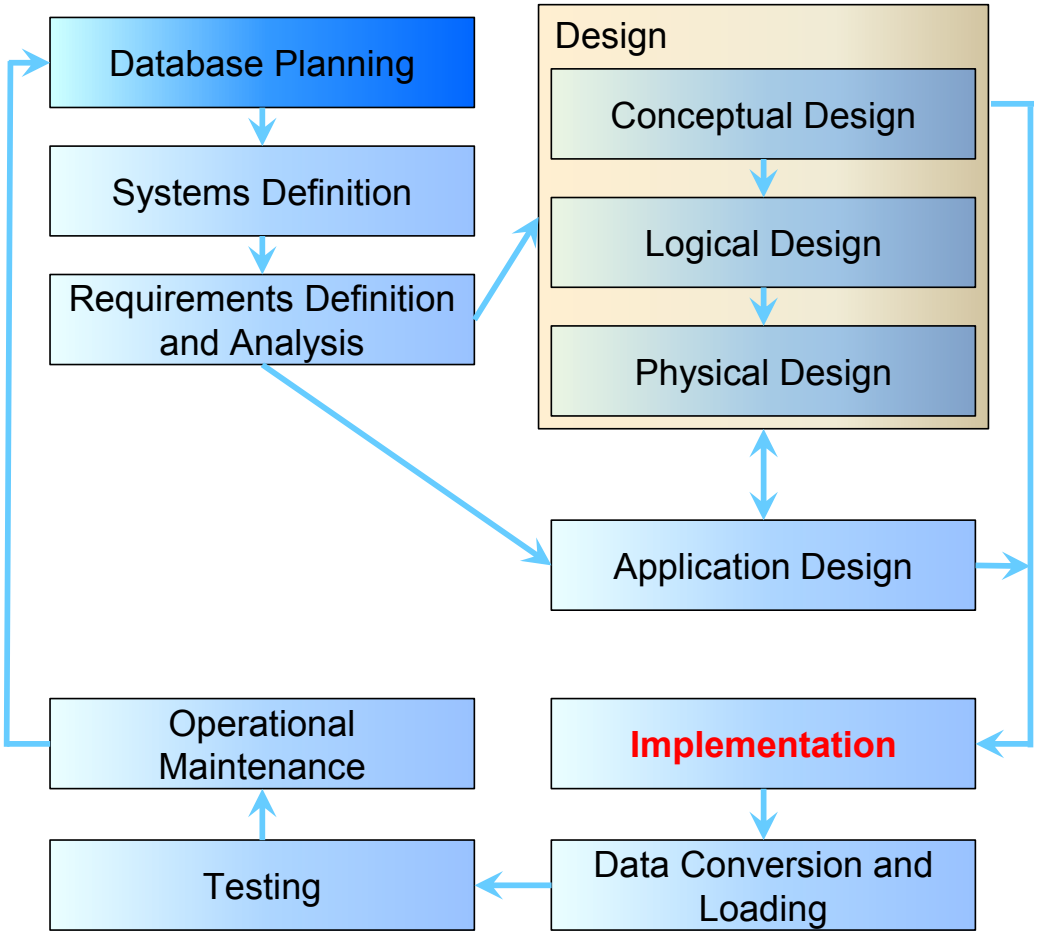
- To De-Normalise or Not (That is the Question)
  - Normalisation
    - A formal method used to validate and improve upon the logical design thus far (which attributes should be grouped together), before proceeding with the physical design.
    - Taught later in the semester
  - De-Normalisation (data warehouse application)
    - At physical design time need to decide how to implement the design – including removing some of the normalisation steps...
    - Benefits
      - Improved database performance
    - Costs
      - Wasted storage space
      - Data integrity / consistency threats

Database Development Lifecycle



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

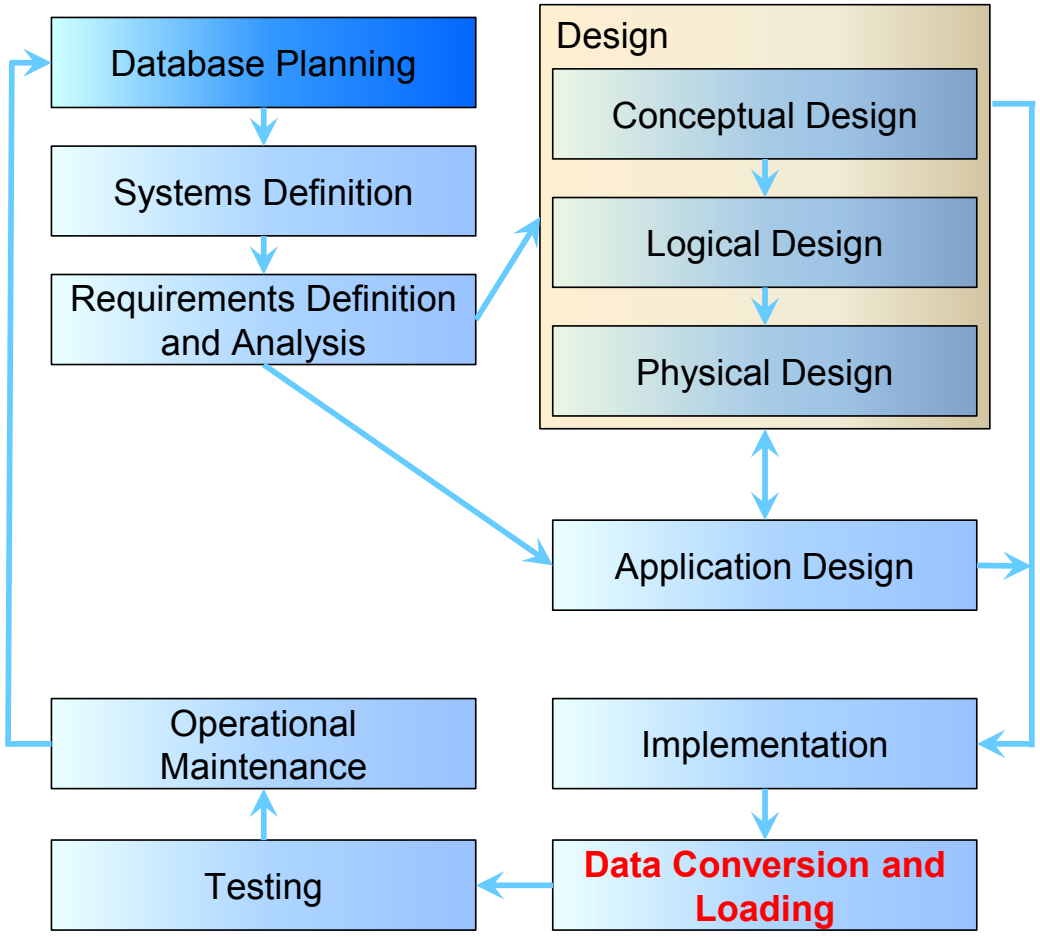
Database Development Lifecycle



- The physical realisation of the database
- Implementation of the design

- Some of the things in implementation are covered as you will be implementing database tables

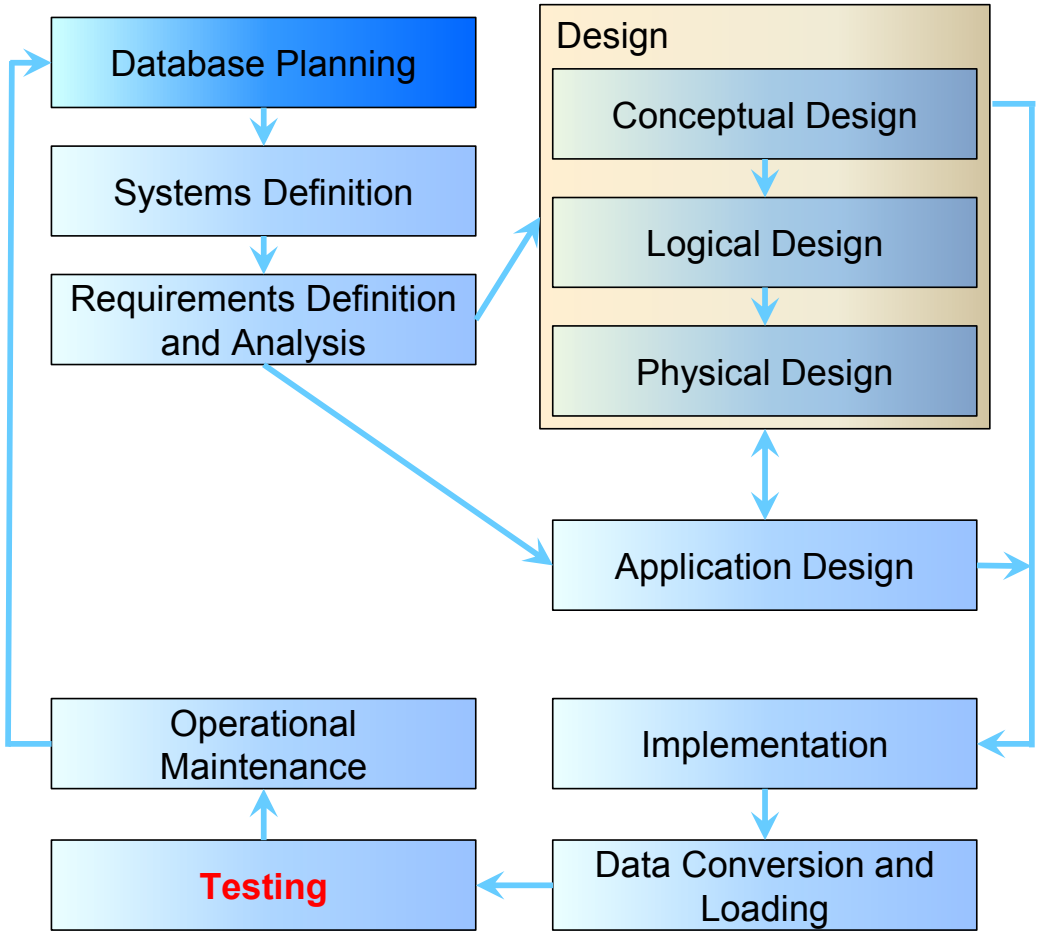
Database Development Lifecycle



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

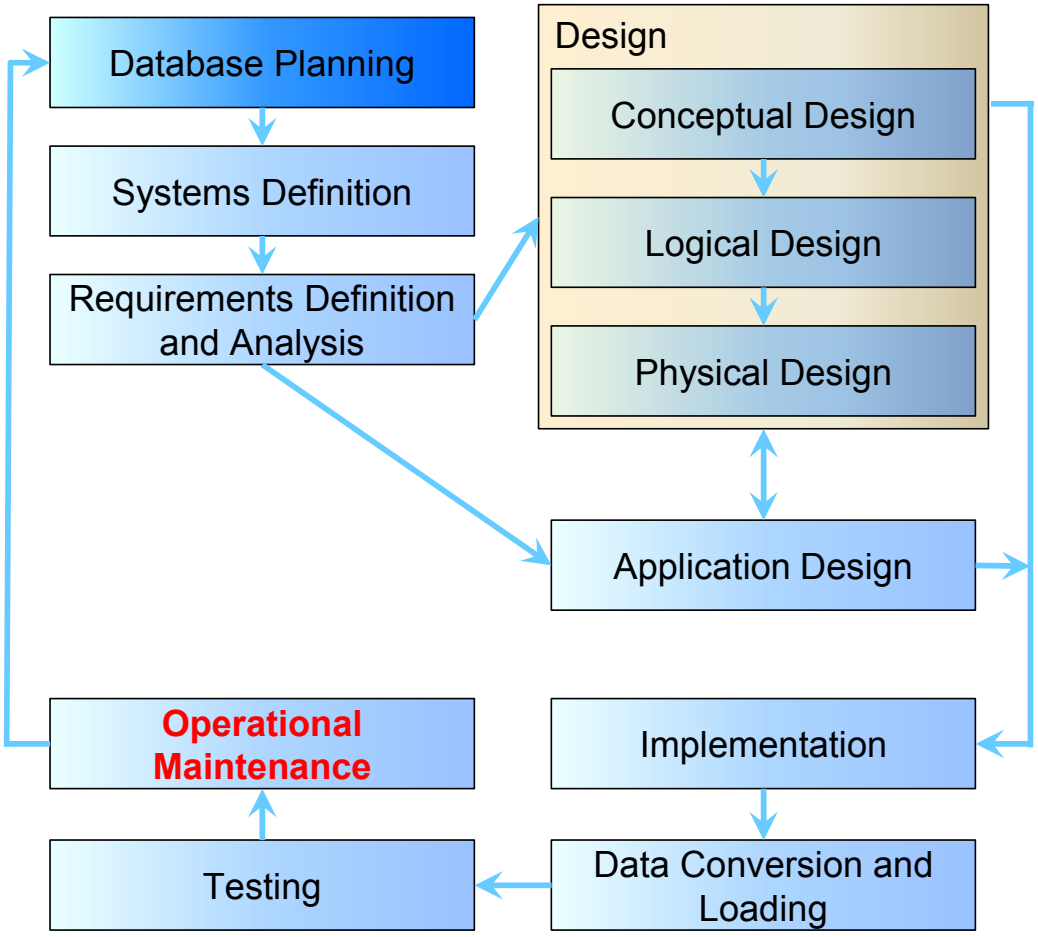
- We give you the data. In a real world situation you would have to do this step – very carefully, very time consuming... Lots of issues around this

Database Development Lifecycle



- Running the database to find errors in the design / setup (both at a physical level and at a logical level)
- Other issues also
  - Performance
  - Robustness
  - Recoverability
  - Adaptability
- Outside scope of the course (slightly) – but you need to test your solutions

Database Development Lifecycle

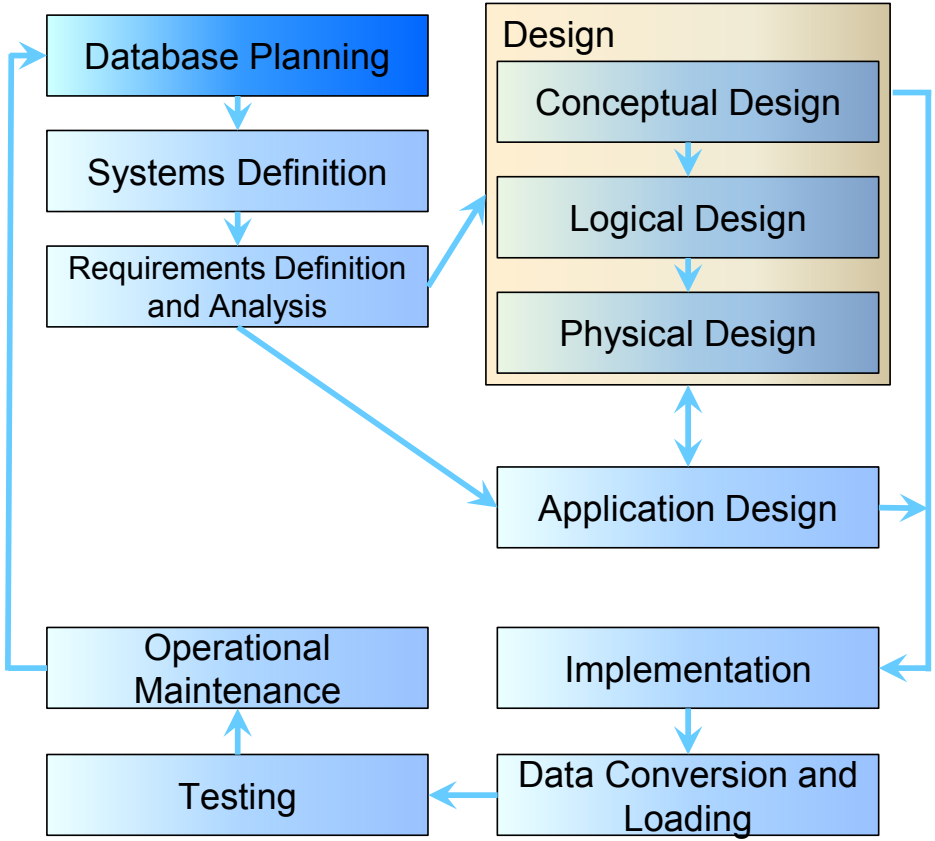


- The process of monitoring and maintaining the database system following its commissioning
- Handling new requirements
- Handling changes to requirements

- Outside scope of the course

Summary

- Discussed the lifecycle of Database Development
- Showed detail of the Modelling stages



What's Examinable?

- Can you discuss the Database Development Lifecycle?
- What is done at each stage of Design?

Next Lecture

- Introduction to Database Design
  - Conceptual design (ER diagrams)

范式：符合某一种级别的关系模式的集合。构造数据库必须遵循一定的规则

第一范式：1NF 是对属性的原子性，要求列不可再分解。

第二范式：2NF 是对记录的惟一性，即实体的惟一性，即不存在部分依赖

⇒：拆表。两表关联使用，消除了数据冗余。

第三范式 3NF 是对字段的冗余性，要求字段没有冗余，即不存在传递依赖。

表：学号，姓名，年龄，学院名称，学院电话。

存在依赖传递：学号 → 学生 → 所在学院 → 学院电话

反范式：有时没有冗余的数据库未必是最好的数据库。

为了提高运行效率，必须降低范式标准。

（以空间换时间）。

范式化 vs 反范式化 → 优：减少表关联，更好地进行索引优化。

（空间换时间） 缺：存在大量冗余数据，数据维护成本高

↓

优：时间换空间

减少了数据冗余，数据表更作操作快，占用存储空间少。

缺：查询时需要多个表关联，查询性能降低