Overview

What is a database?

- a model of reality!

  why use models?

  when to use a Database Management System (DBMS) to make a Model?

Why Use Models?

Some Observations:

- models can be useful when we want to examine or manage part of the real world
- the costs of using a model are often considerably lower than the costs of using or experimenting with the real world itself

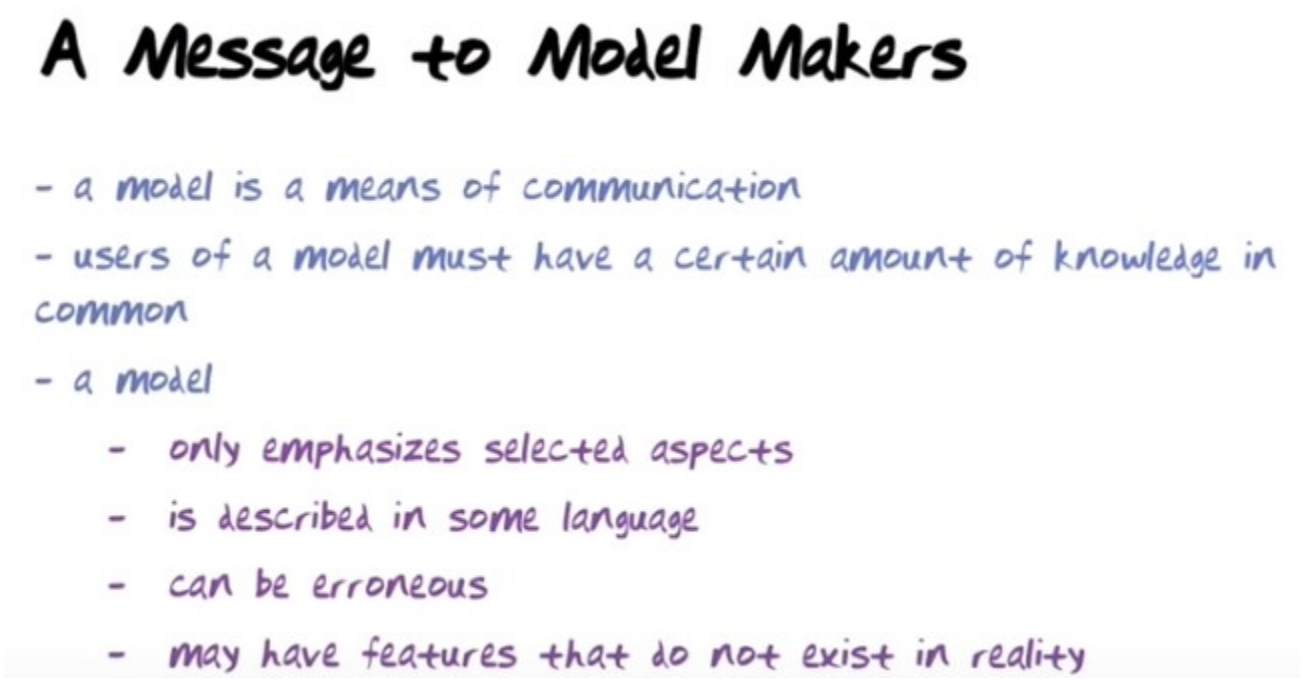
Example:

- a map
- others include:

  model of US economy

  +sunami warning system

  +traffic simulation



What is a database management system?

- a) A collection of data stored on a computer
- b) A collection of related data representing facts that have meaning in the real world
- c) A software system allowing users to create and maintain a database
- d) A software system allowing users to define web-portal interfaces that interact with data files

Piazza 上有人對以上答案表示懷疑, instructor's reply:

You might find everyone defines DBMS is a slightly different way (you will notice the same pattern in IOS or AOS where you have to define "operating system").

Here are relevant snippets from the book (Elmasri, Navathe):
"A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know"
"A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications."

(As for the Knowledge Checks on Udacity, we are still fiddling with them. At the moment, Knowledge Check 1 is incorrectly coded. We will correct it and get back to you)

DATABASE MANAGEMENT SYSTEM (DBMS)

TO USE or NOT TO USE

TO USE

- ✓ data intensive apps
- ✓ persistent storage of data
- ✓ centralized control of data
- ✓ control of redundancy
- ✓ control of consistency and integrity
- ✓ multiple user support
- ✓ sharing of data
- ✓ data documentation
- ✓ data independence
- ✓ control of access and security
- ✓ backup and recovery

NOT TO USE

- ✗ the initial investment in hardware, software, and training is too high
- ✗ the generality is not needed

  Example: the overhead for security, concurrency control, and recovery is too high
- ✗ data and applications are simple and stable
- ✗ real-time requirements cannot be met by it
- ✗ multiple user access is not needed

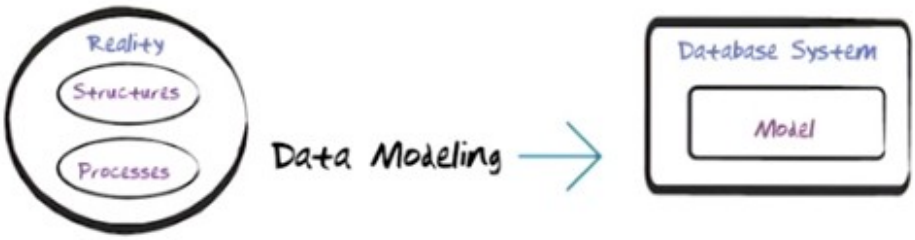
Outline of Major Topics

Major Topics:

- data modeling
- process modeling
- database efficiency

Data Modeling

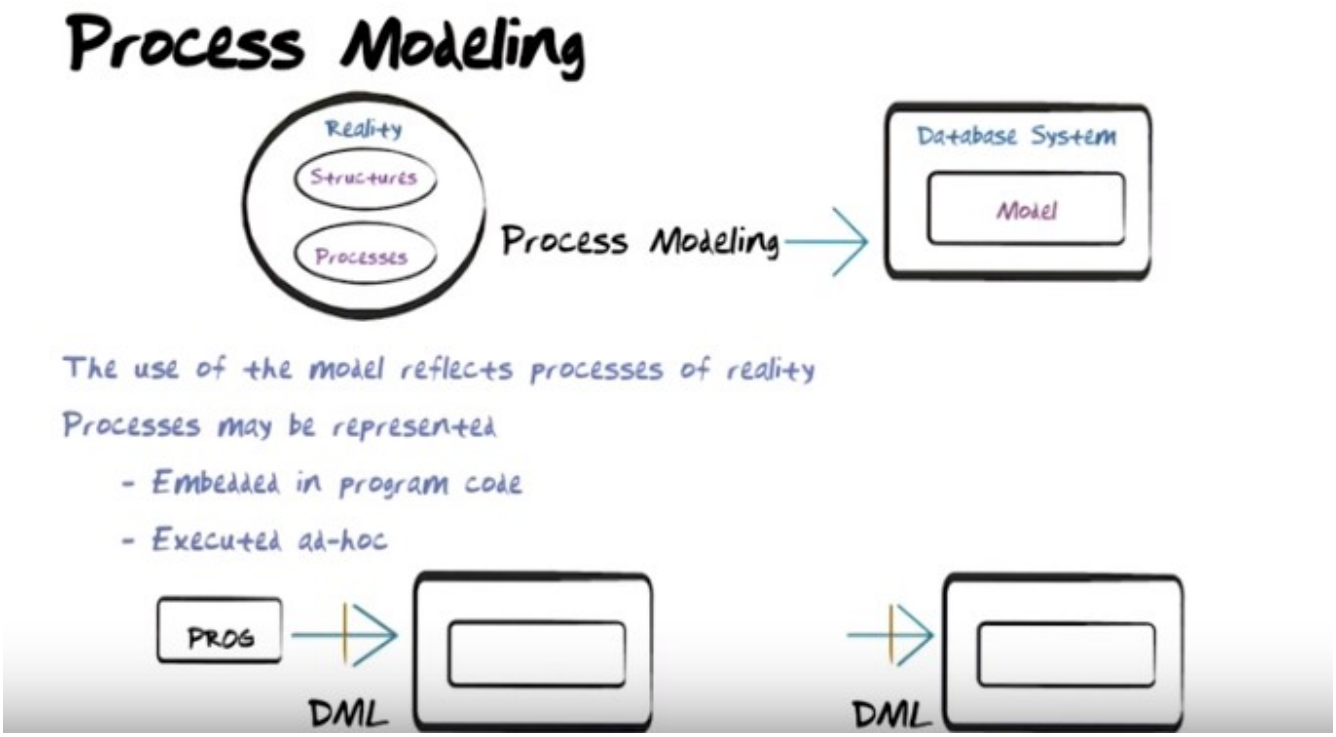
The model represents a perception of structures of reality



The data modeling process is to fix a perception of structures of reality and represent this perception

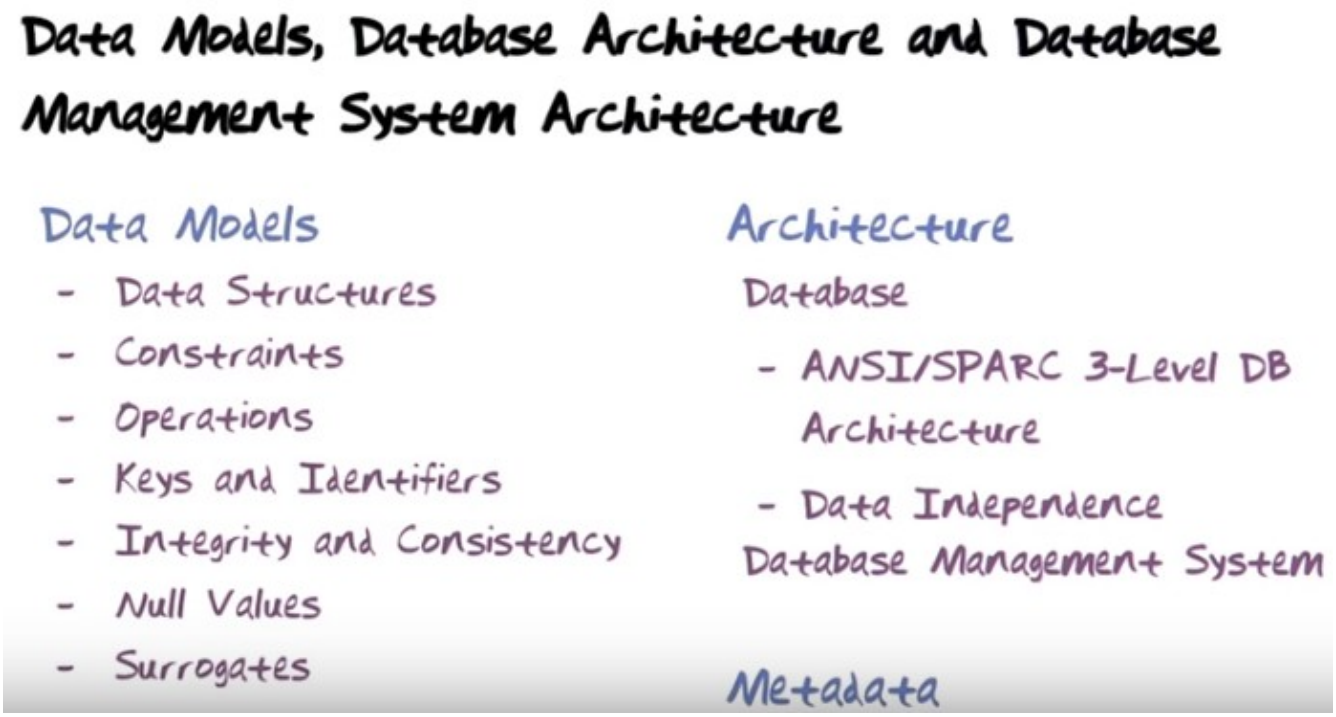
In the data modeling process we select aspects and we abstract

Database people use two different languages for this. One language is the extended entity-relationship model which is good for fixing a perception of stuctures of reality, second the relational model is good for representing this perception inside a database.



In the above figure, “Embeded in program code” means to embed data manipulation language (DML) code inside a program. “Executed ad-hoc” (ad-hoc 字面是特別的意思) means to use the data manipulation language directly to issue ad-hoc queries on the database.

The data manipulation language we're goint to use in this course is gonna be SQL. You're gonna spend major amount of time in the course learning how to build programs likes this (left bottom picture of the above figure, 已驗證) and how to embed in the SQL statement to manipulate maintain the database.



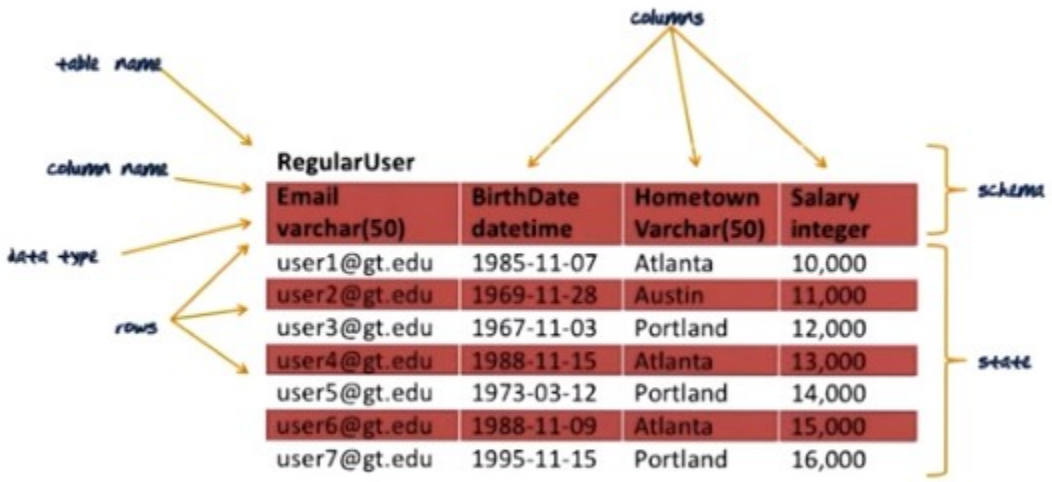
(the figure above) A data model is not the same as a model of data. We defined a database to be a model of structures of reality. And data model (見下段) is the tool or the formalism that we use to create such a model.

Let me mention three extremes of data models. The first one is the entity-relationship model. We will use this model to fix a perception of reality in this course. Second is the relational model which we will use to implement that model in a database management system.

以下分別講上上圖中的 Data Structures, Integrity Constraints, 和 Operations.

An example of data model. Let me use the relation model. In the relational model, data is represented in tables. The number of columns is also called the degree of the table.

Relational Model - Data Structures



schema: stable over time

state: dynamic; reflects the state of reality

Relational Model - Constraints

Constraints express rules that cannot be expressed by the data structures alone:

- Email's must be unique
- Email's are not allowed to be NULL
- BirthDate must be after 1900-01-01
- Hometown must be cities in the US

RegularUser

Email varchar(50)	BirthDate datetime	Hometown Varchar(50)	Salary integer
user1@gt.edu	1985-11-07	Atlanta	10,000
user2@gt.edu	1969-11-28	Austin	11,000
user3@gt.edu	1967-11-03	Portland	12,000
user4@gt.edu	1988-11-15	Atlanta	13,000
user5@gt.edu	1973-03-12	Portland	14,000
user6@gt.edu	1988-11-09	Atlanta	15,000

Data Model - Operations

Operations support change and retrieval of data:

```
insert into RegularUser
(user11@ugedu, 1992-10-22, Atlanta, 12,500);

select Email, BirthDate
    from RegularUser
     where Hometown='Atlanta' and Salary>12,000;
```

RegularUser

Email varchar(50)	BirthDate datetime	Hometown Varchar(50)	Salary integer
user1@gt.edu	1985-11-07	Atlanta	10,000
user2@gt.edu	1969-11-28	Austin	11,000
user3@gt.edu	1967-11-03	Portland	12,000
user4@gt.edu	1988-11-15	Atlanta	13,000
user5@gt.edu	1973-03-12	Portland	14,000
user6@gt.edu	1988-11-09	Atlanta	15,000

Keys and Identifiers

Keys are uniqueness constraints

- Making Email the primary key in RegularUser will force all Emails to be unique in RegularUser

RegularUser

Email	BirthDate	Hometown
user1@gt.edu	1985-11-07	Atlanta
user2@gt.edu	1969-11-28	Austin
user3@gt.edu	1967-11-03	Portland
user4@gt.edu	1988-11-09	Atlanta
user5@gt.edu	1973-03-12	Portland
user6@gt.edu	1988-11-09	Atlanta

Which of the following might result in poor consistency in a database?

a) Storing Street, City, State and ZIP code as well as an address string in another field to make printing address labels for customers easier

b) Storing both gender and first name of customers

c) Storing both Date of Birth and Age of customers to save computation time for Age

d) a) and c)

Integrity(正直,完整) and consistency are true highly desirable properties of databases.

Integrity and Consistency

以上是講 Integritiy. 以下是講 Consistency, 其中 user1 的 CurrCity 和 Address 不 consistent, user4 也是.

Integrity and Consistency

Integrity: does the database reflect reality well?

Consistency: is the database without internal conflicts?

RegularUser

Email	BirthDate	Name	CurrCity
user1@gt.edu	50-11-07	Leo Mark Christensen	Atlanta
user2@gt.edu	69-11-28	Elizabeth Smith	Austin
user3@gt.edu	67-11-03	Joanne Fulton	Portland
user4@gt.edu	81-11-09	Louise Mark Christensen	Atlanta
user5@gt.edu	73-03-12	Jennifer Liu	Portland
user6@gt.edu	11-11-09	Edward Booth	Atlanta

User

Email	Address
user1@gt.edu	123 Kent Rd., Roswell
user2@gt.edu	456 Ferst St., Austin
user3@gt.edu	12 Virginia Av., Portland
user4@gt.edu	789 1st St., Roswell
user5@gt.edu	123 Kahn Rd., Portland
user6@gt.edu	654 MLK Blvd., Atlanta

A surrogate:

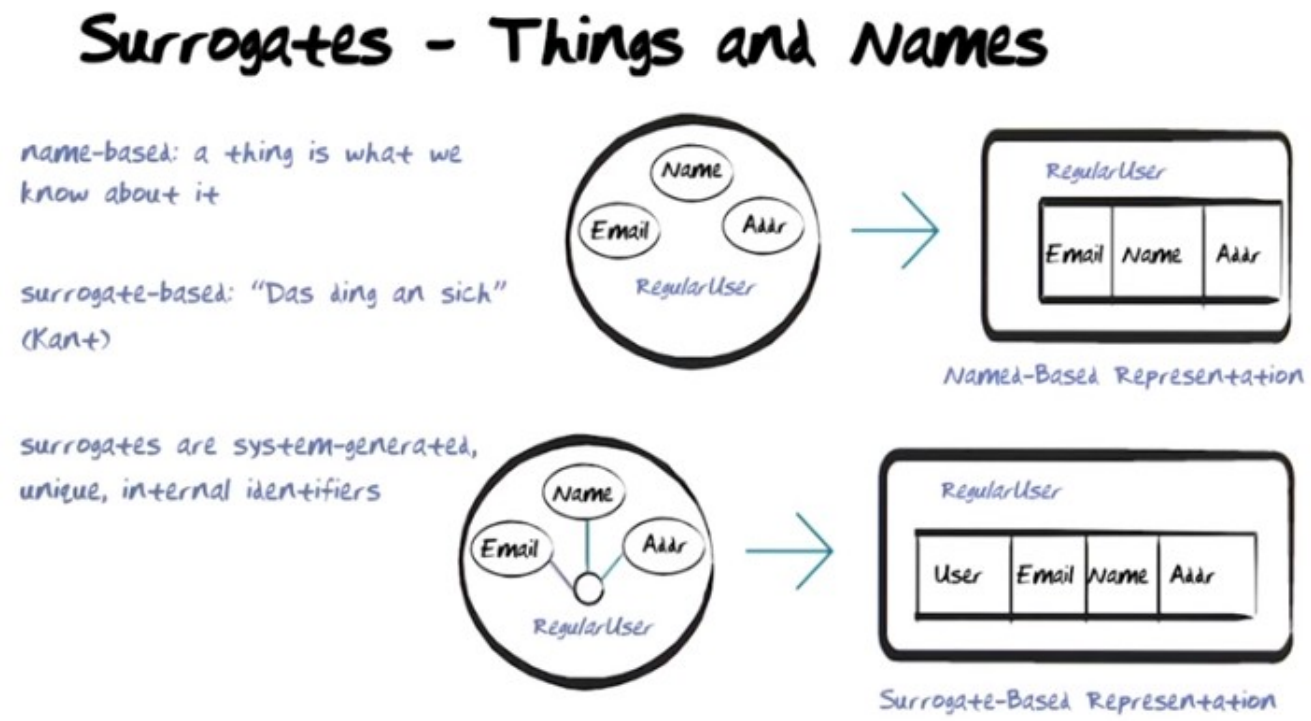
- a) Is a system generated artificial identifier for an entity
- b) Represents an entity of the real world inside the database
- c) Is immutable by the application programs
- d) All of the above

NULL Values

RegularUser

Email	BirthDate	Hometown	LastName	MaidenName	Sex	Salary
user1@gt.edu	1985-11-07	Seattle	Christensen	INAPPLICABLE	M	10,000
user2@gt.edu	1969-11-28	Austin	Smith	UNKNOWN	F	11,000
user3@gt.edu	1967-11-03	San Diego	Fulton	Jones	F	12,000
user4@gt.edu	1988-11-15	San Francisco	Russo	Christensen	F	13,000
user5@gt.edu	1973-03-12	San Francisco	Liu	UNKNOWN	F	14,000
user6@gt.edu	1988-11-09	San Diego	Booth	INAPPLICABLE	M	15,000
user7@gt.edu	1995-11-15	San Francisco	Navathe	INAPPLICABLE	M	16,000
user8@gt.edu	1968-08-12	College Park	Zelkowitz	INAPPLICABLE	M	17,000

(the above figure) Null value unknown is OK. It would be desirable to have a value but it's OK. Null value inapplicable is not OK. It means that we have designed a table in such a manner that several of the rows are not ganna have an applicable value for some of the columns. Not very good. How can that happen? It could happen, for example, if you lay out a table to reflect catch all forms. So be very careful. Catch all forms are great in reality, simplies which forms you have to keep in stock and which forms people have to fill out. Think for example about your favorite form in the world u1048, as a lot of fields on it. Some of them apply to you, some of them do not apply to you. Just because a catch all forms has a number of fields on it does not mean that you desiged one table with all thoes fields on it.



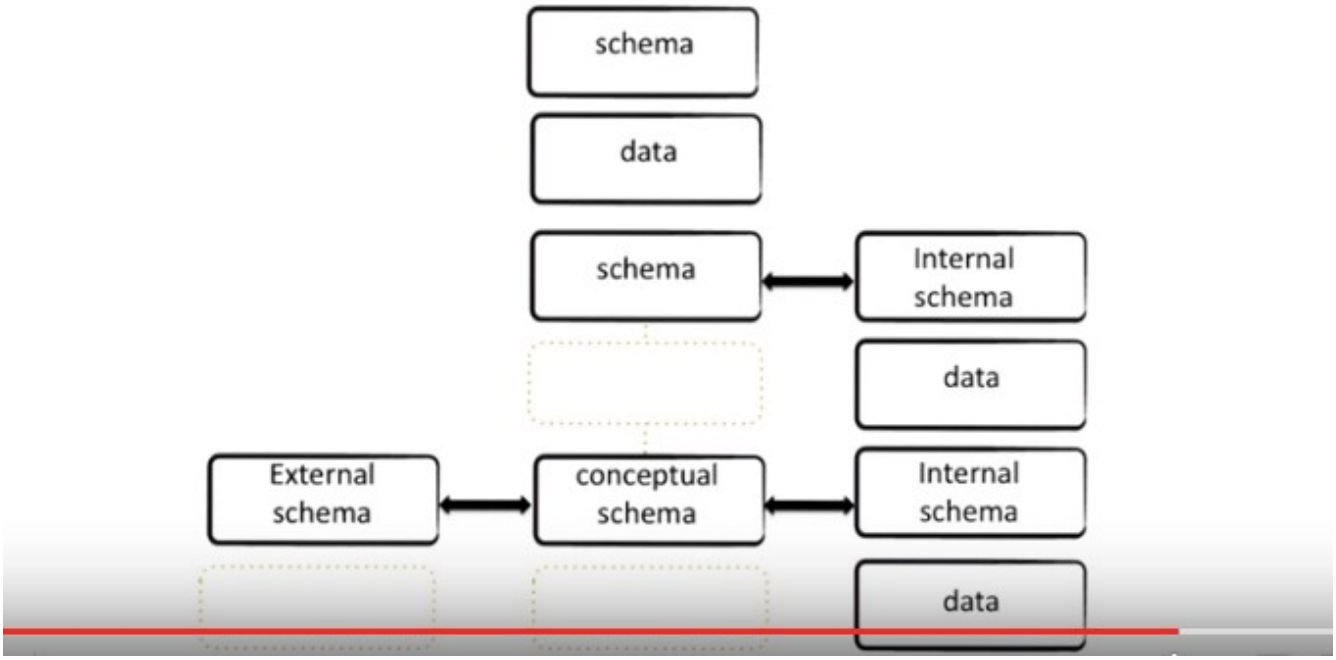
Surrogate: 代理. 上圖的意思是, 用 name-based representation 不好. 因為名字可能會變. 用 User 就好多了. Das Ding an sich, a phrase which literally means “The thing in itself”.

Let's now turn our attention to architecture. We will be talking about the architecture of database and the architecture of the database management system that used to create and maintian the database. The architecture we're gonna look at is called the ANSI/SPARC 3-Level architecture. A database is a model of structures of reality. It's divided into schema(輪廓) and data.

ANSI/SPARC 3-Level DB Architecture: Separating Concerns

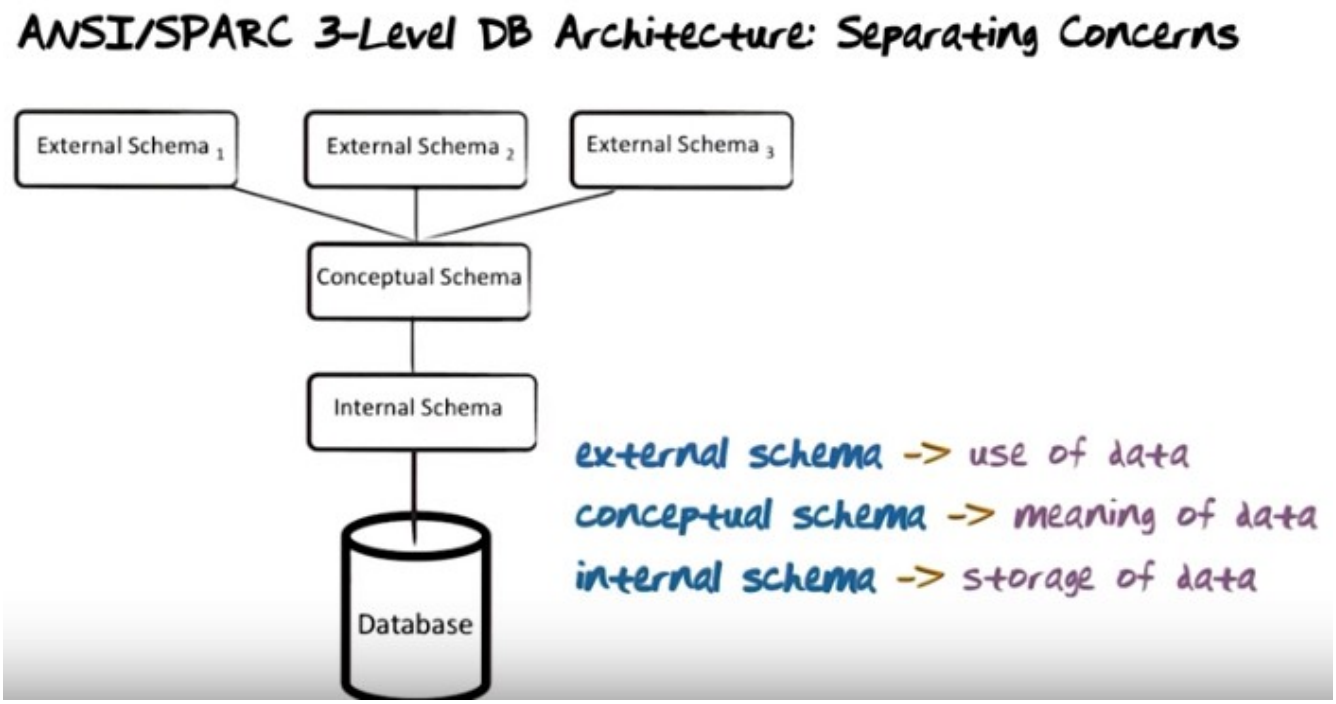
- a database is divided into schema and data
- the schema describes the intension (types)
- the data describes the extension (data)
  - Why? Effective! Efficient!

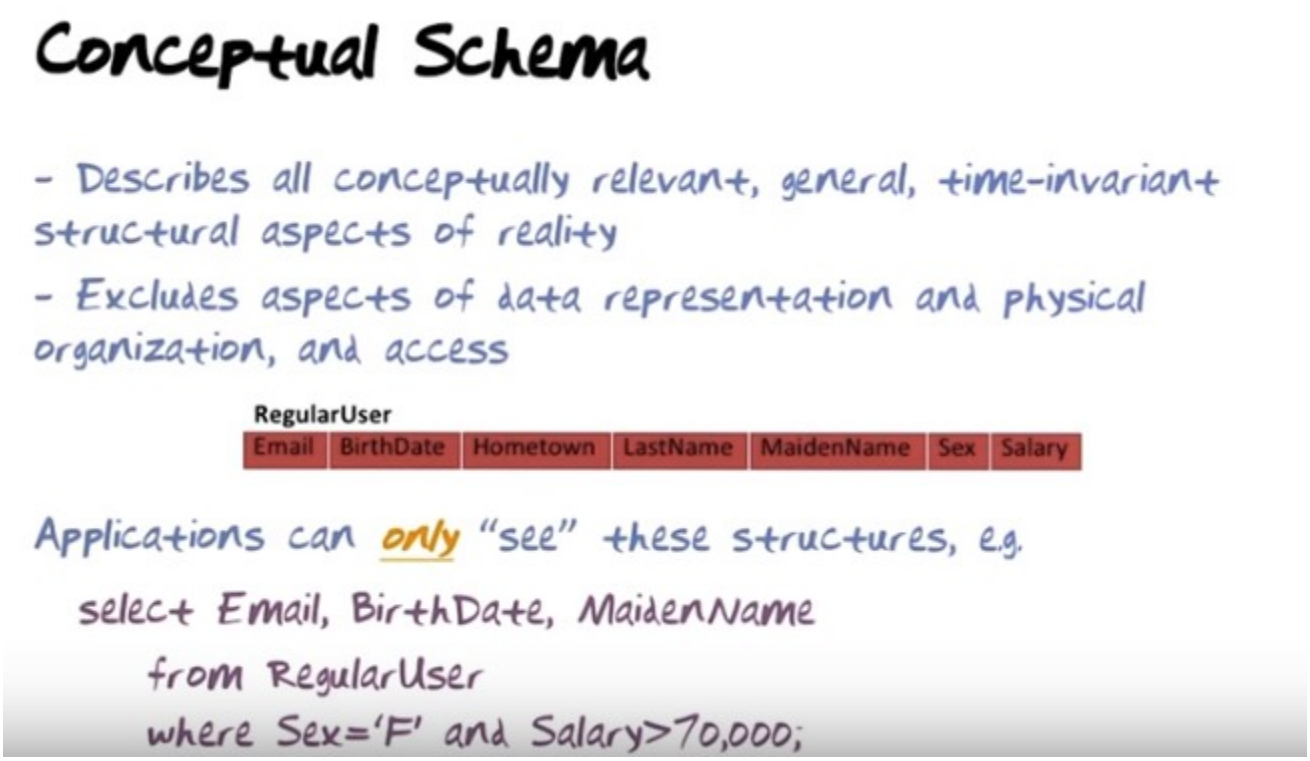
ANSI/SPARC 3-Level DB Architecture: Separating Concerns



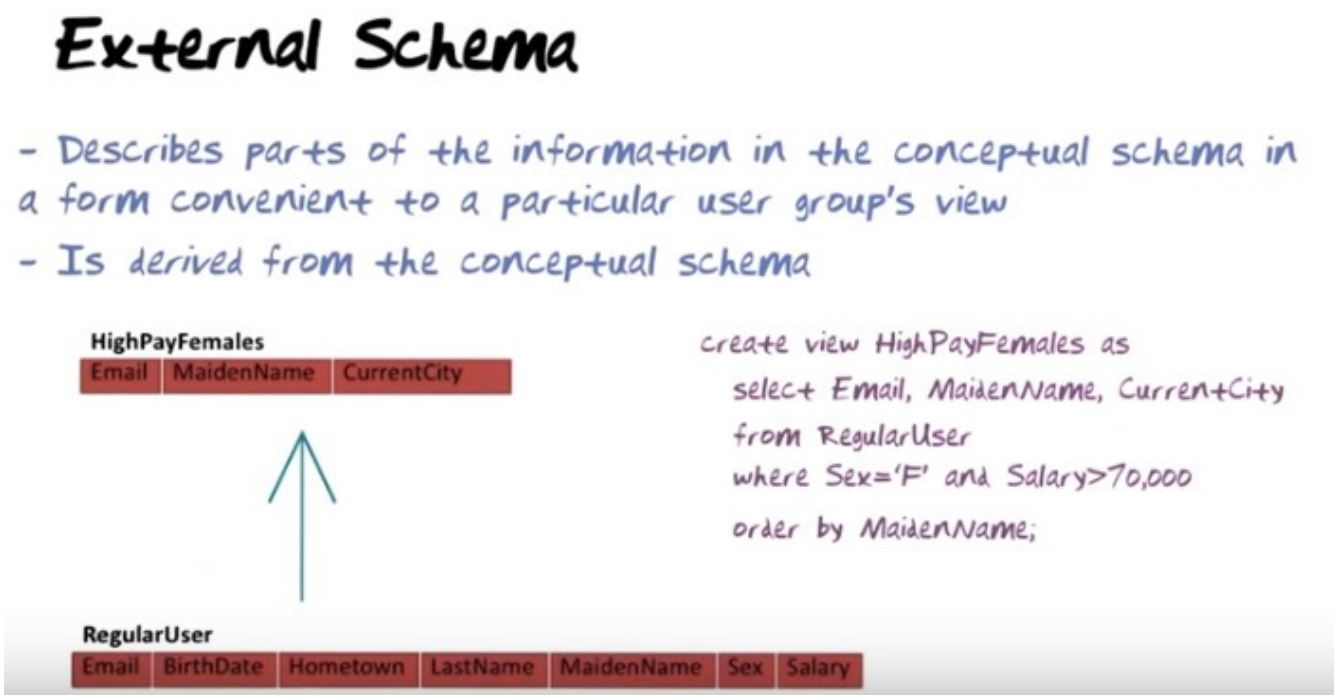
(the above figure) The separation into a single schema and data however is not the end of it. If the database only consisted of schema and data, then the schema would have to describe aspects of what the meaning of data is, how it's used and how it's internally organized. This would mean that queries against the database through their schema would be able to reference how the data is physically organized. The implication of that is, if we decide to change how data is physically organized then the

application of the database would be affected. Therefore the ANSI/SPARC 3-Level architecture separates out aspects of how data is physically organized into what is known as an internal schema, and 'data now sits on the internal schema' starts their database access. It will appear that it goes through the schema to access data, but instead what happens is it goes throgh the schema and the request or change is translated to one at the internal schema level and data is accessed and the answer is send back. The benefit of this is that now it is actually possible to change the way data is stored without affecting the applications that run against the schema. However, a database is used by many different applications with different needs to display data. Therefore for each need for each application of the database we could create an external schema. Therefore the ANSI/SPARC 3-Level prescribes a third-level named the external schema level that separates out aspect of how data is used by individual applications. So now you have applications running against external schema that represents data into preferred format for those applications. Requests through the external schema are then translated to requests through a conceptual schema, translated to request through the internal schema. Data is accessed and the responses translated up and presented to the application in the format that it needs. (See the figure below) Turn this illurstratoin 90 degrees and you get to the 3-Level ANSI/SPARC database architecture that people normally see depicted. You have a conceptual schema in the middle, you have an internal schema implementing that, and you hava a number of external schema, one for each major application.





(the above figure, 注意圖中第二點是 excludes, 不是 includes). The only thing that's visible to the query language is this table and a column in it. So you could, for example, write a query to select email, birthdate and maiden name from regular user where Sex = 'F' and Salary > 70,000. You can not in that query include anything about how the results are displayed other than in this order or how the data is accessed.



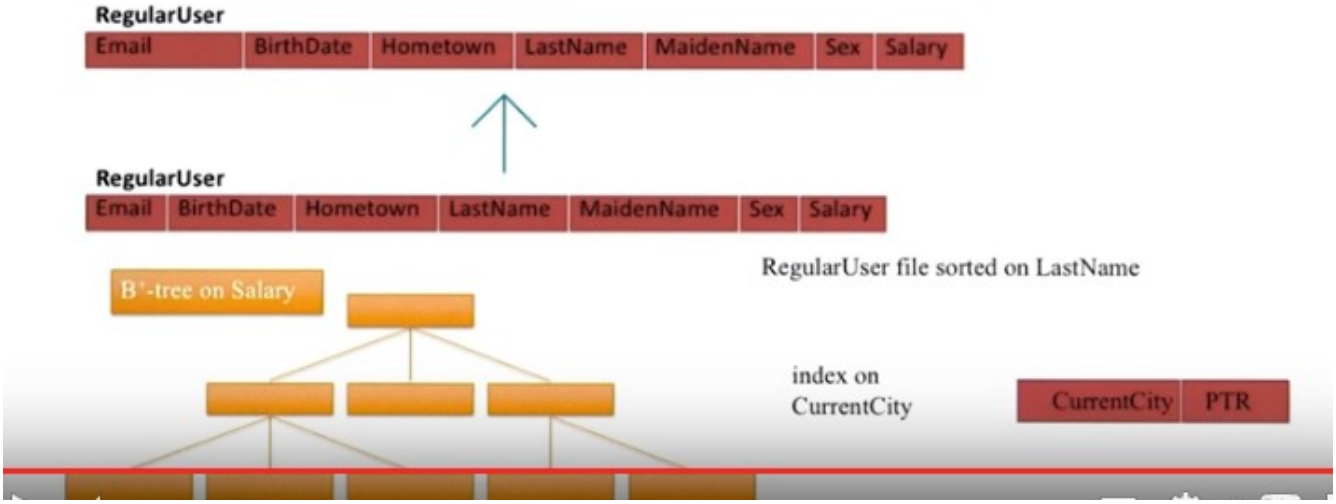
(the above figure) Here is an example of a virtual table that exists in an extenal schema. The name of the virtual table or the view is HighPayFemales.

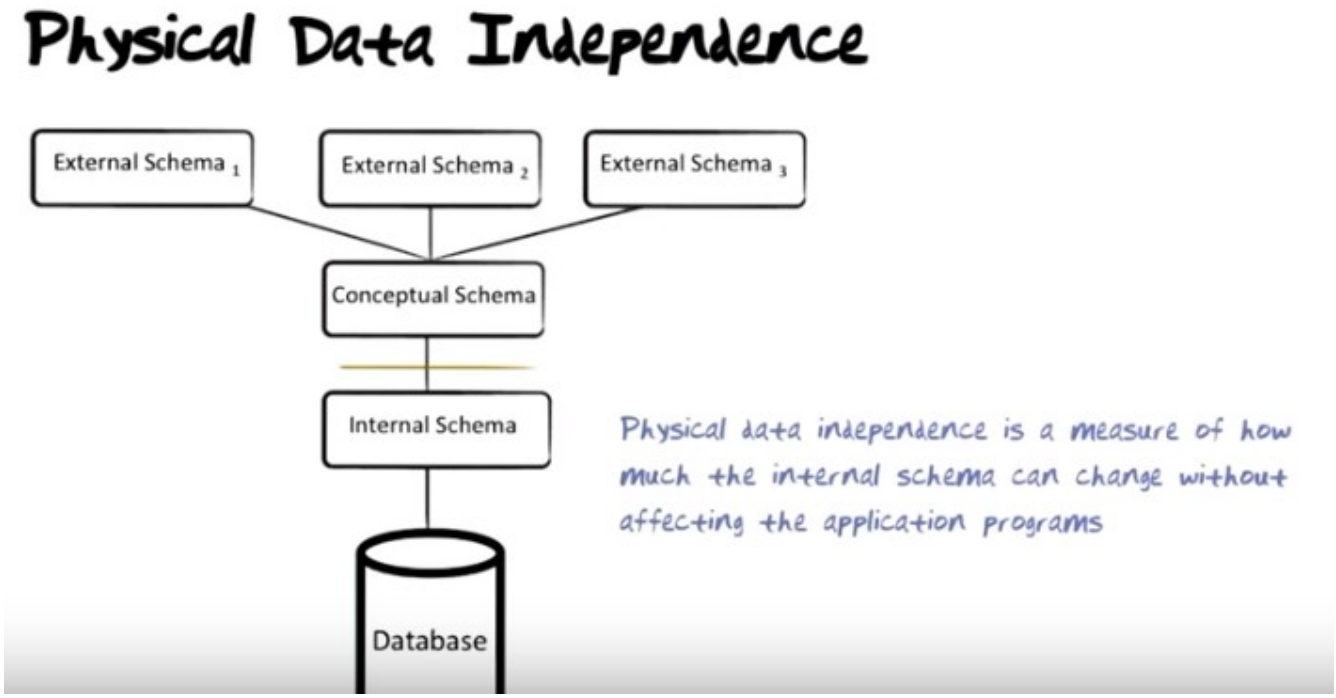
What does physical data independence allow the database administrator to do?

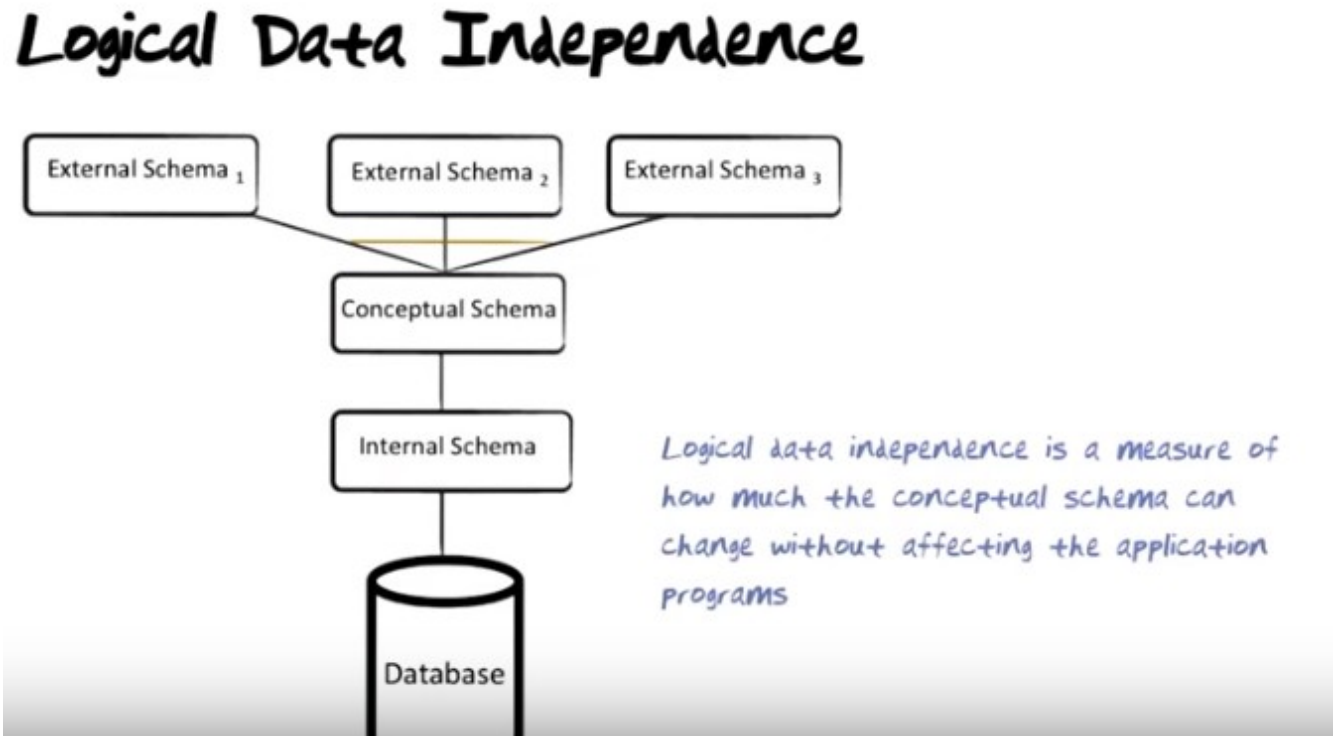
- a) Change the conceptual schema without changing the internal schema
- b) Change the internal schema without changing the conceptual schema
- c) Change the external schema without changing the conceptual schema
- d) Change the application programs without changing the internal schema

Internal Schema

- Describes how the information described in the conceptual schema is physically represented to provide the overall best performance



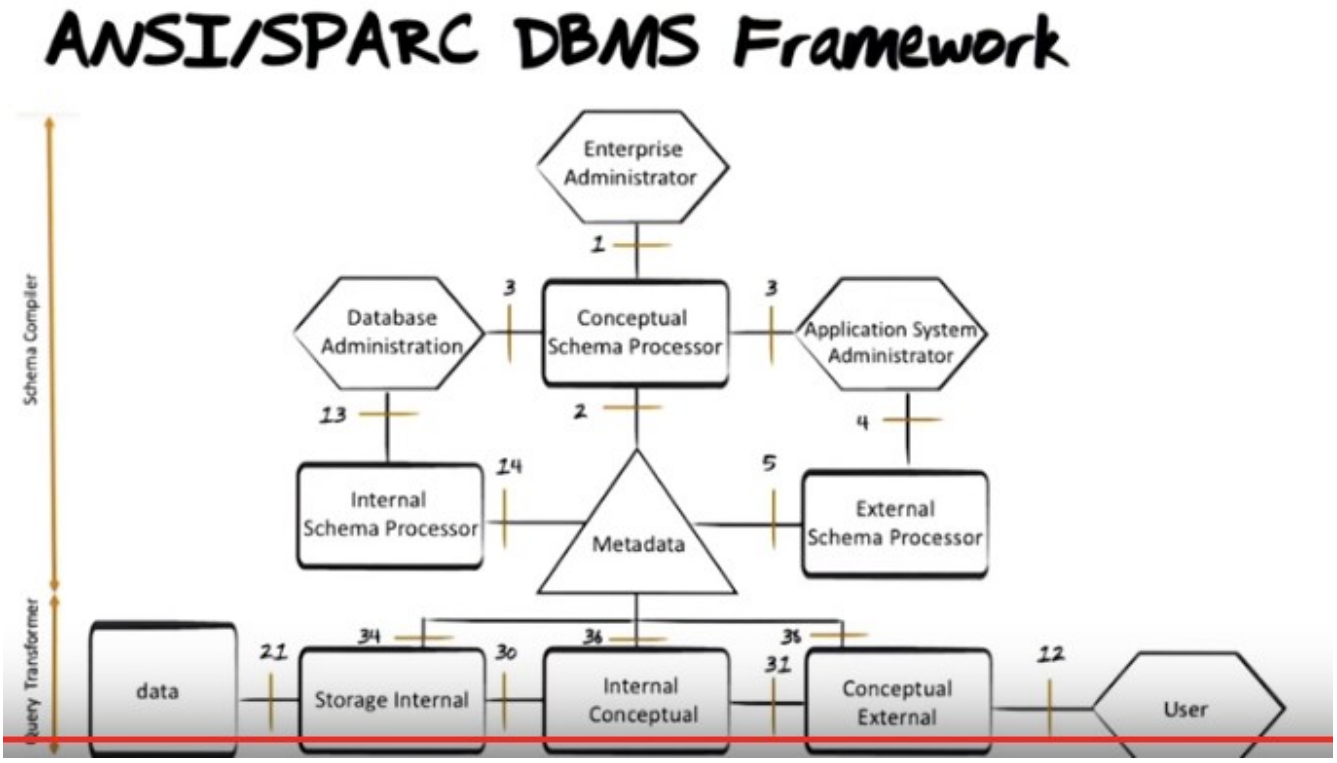




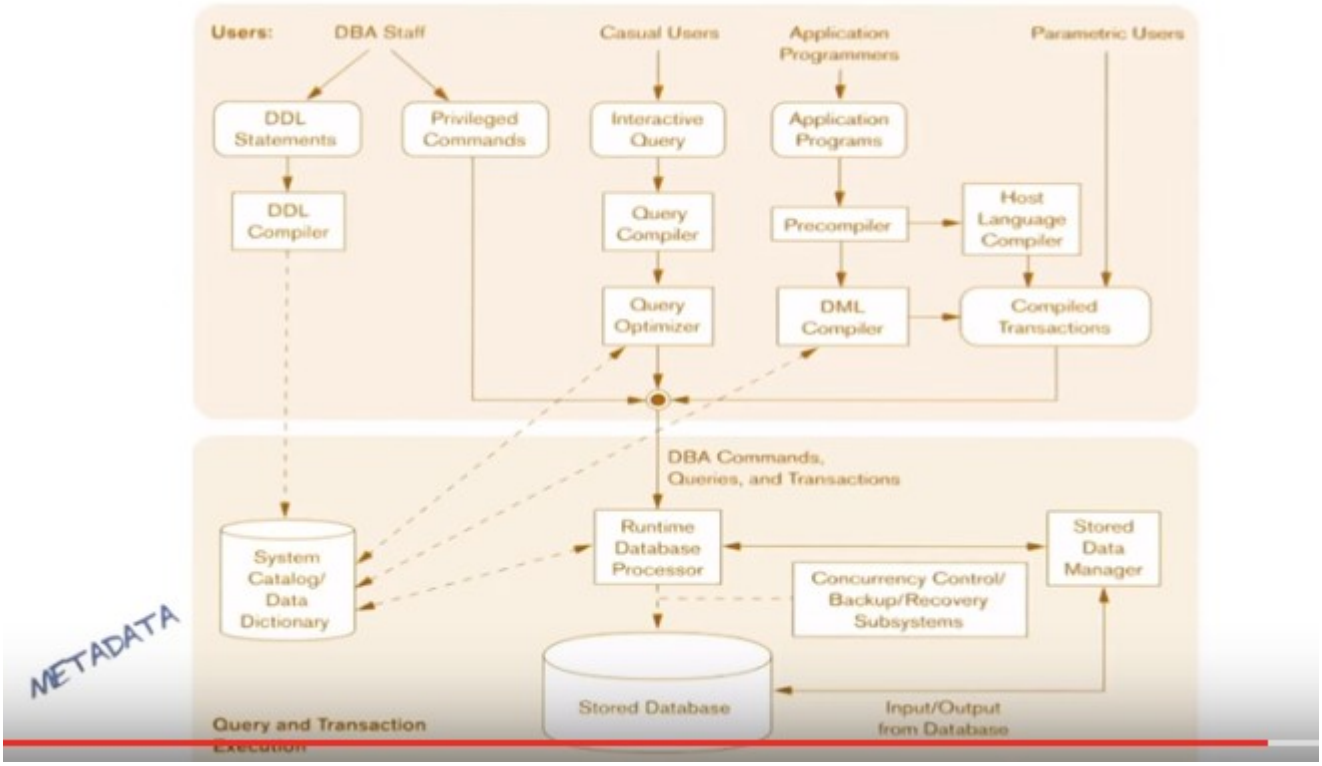
(the above figure). It is more diffcult to provide logical data independence than it is to provide physical data independence. The reason is that the external schema against which the applications are written, those external schema are logically derived from the conceptual schema. So needless to say, if you have an application that accesses a table in an external schema, and you go change that table in the conceptual schema, then it is possible that your application will be affected.

Which of the following is/are important characteristic(s) of a database approach?

- a) Information describing the database (ie, meta-data) is stored in a catalog
- b) Different users may see different views of the database
- c) Data may be shared among many users
- d) All of the above



(the figure below) Here is a somewhat different rendition(描寫) of what I just described to you.



(the figure below) Metadata is essential for making everything happen in a database (沒說甚麼是 metadata). One distinguish between two kinds of metadata. One is system metadata, the other one is business metadata.