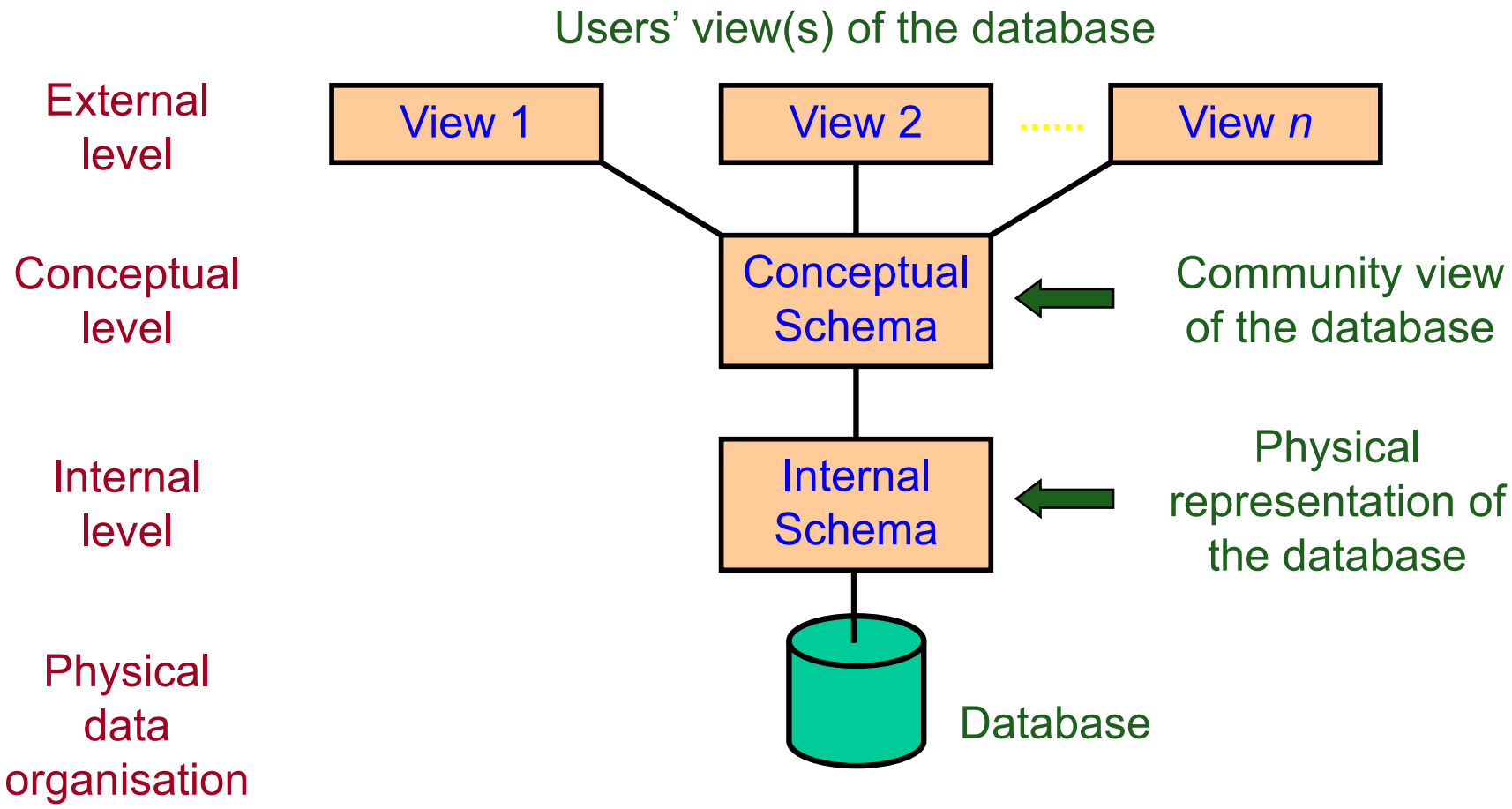
CSCU9B3
Database Principles and Applications

The ANSI-SPARC Architecture

The ANSI-SPARC Architecture

- The architecture of most commercial DBMSs is based on the ANSI-SPARC architecture (1975).
  - American National Standards Institute (ANSI)
  - Standards Planning And Requirements Committee (SPARC)
- Although this never became a formal standard, it is useful to help understand the functionality of a typical DBMS.
- The ANSI-SPARC model of a database identifies three distinct levels at which data items can be described.
- These levels form a three-level architecture comprising:
  - an *external* level,
  - a *conceptual* level, and
  - an *internal* level.

The Three-Level Architecture - I



The Three-Level Architecture - II

- The objective of the three-level architecture is to separate the users' view(s) of the database from the way that it is physically represented. This is desirable for the following reasons:

1. It allows independent customised user views.
   - Each user should be able to access the same data, but have a different customised view of the data. These should be independent: changes to one view should not affect others.
2. It hides the physical storage details from users.
   - Users should not have to deal with physical database storage details. They should be allowed to work with the data itself, without concern for how it is physically stored.

- More ...

The Three-Level Architecture -III

3. The database administrator should be able to change the database storage structures without affecting the users' views.
   - From time to time rationalisations or other changes to the structure of an organisation's data will be required.

4. The internal structure of the database should be unaffected by changes to the physical aspects of the storage.
   - For example, a changeover to a new disk.

5. The database administrator should be able to change the conceptual or global structure of the database without affecting the users.
   - This should be possible while still maintaining the desired individual users' views.

The External Level

- The external level represents the user's view of the database.
  - It consists of a number of different views of the database, potentially one for each user.
- It describes the part of the database relevant to a particular user.
  - For example, large organisations may have finance and stock control departments.
  - Workers in finance will not usually view stock details as they are more concerned with the accounting side of things, for example.
  - Thus, workers in each department will require a different user interface to the information stored in the database.
- Views may provide different representations of the same data.
  - For example, some users might view dates in the form (day/month/year) while others prefer (year/month/day).
- Some views might include derived or calculated data.
  - For example, a person's age might be calculated from their date of birth since storing their age would require it to be updated each year.

The Conceptual Level

- The conceptual level describes what data is stored in the database and the relationships among the data.
- It is a complete view of the data requirements of the organisation that is independent of any storage considerations.
- The conceptual level represents:
  - All entities, their attributes, and their relationships.
  - The constraints on the data.
  - Security and integrity information.
- The conceptual level supports each external view, in that any data available to a user must be contained in, or derivable from, the conceptual level.
- The description of the conceptual level must not contain any storage-dependent details.

The Internal Level

- The internal level covers the physical representation of the database on the computer (and may be specified in some programming language).
- It describes how the data is stored in the database in terms of particular data structures and file organisations.
- The internal level is concerned with:
  - Allocating storage space for data and indexes.
  - Describing the forms that records will take when stored.
  - Record placement. Assembling records into files.
  - Data compression, security and encryption techniques.
- The internal level interfaces with the OS to place data on the storage devices, build the indexes, retrieve the data, etc.
- Below the internal level is the physical level which is managed by the OS under the direction of the DBMS. It deals with the mechanics of physically storing data on a device such as a disk.

Differences between the Levels

External View 1

Sno	Fname	Lname	Age	Salary

External View 2

StaffNo	Lname	Bno

Conceptual Level

StaffNo	Fname	Lname	DOB	Salary	BranchNo

Internal Level

```
Struct STAFF {
  int StaffNo;
  int BranchNo;
  char Fname[15];
  char Lname[15];
  struct date DateOfBirth;
  float Salary;
  struct STAFF *next; // pointer to next record
};
```

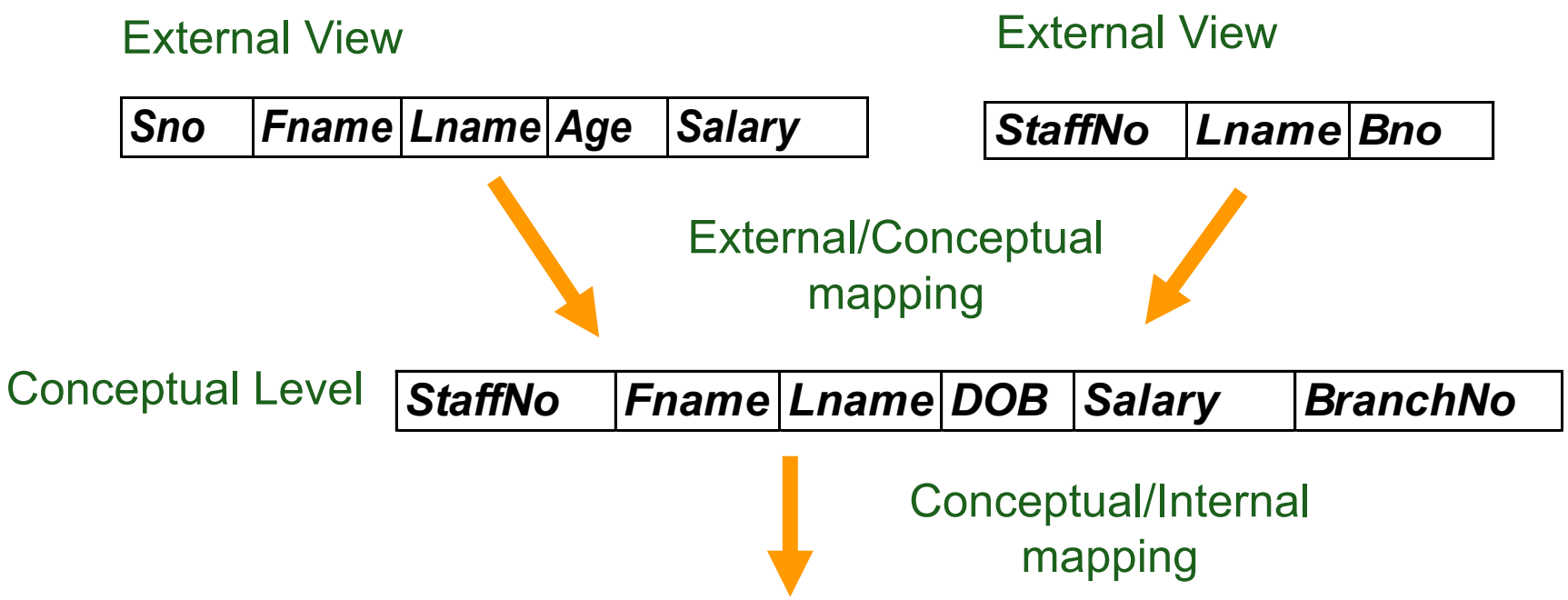
Database Schemas

- The overall description of a database is called the *database schema.*
- There are three different types of schema corresponding to the three levels in the ANSI-SPARC architecture.
- The *external schemas* describe the different external views of the data.
  - There may be many external schemas for a given database.
- The *conceptual schema* describes all the data items and relationships between them, together with integrity constraints (later).
  - There is only one conceptual schema per database.
- At the lowest level, the *internal schema* contains definitions of the stored records, the methods of representation, the data fields, and indexes.
  - There is only one internal schema per database.

Mapping Between Schemas

- The DBMS is responsible for mapping between the three types of schema (i.e. how they actually correspond with each other).
- It must also check the schemas for consistency.
  - Each external schema must be derivable from the conceptual schema.
- Each external schema is related to the conceptual schema by the *external/conceptual mapping.*
- This enables the DBMS to map data in the user's view onto the relevant part of the conceptual schema.
- A *conceptual/internal mapping* relates the conceptual schema to the internal schema.
- This enables the DBMS to find the actual record or combination of records in physical storage that constitute a logical record in the conceptual schema.

Example of the Different Levels



Internal Level

```
Struct STAFF {
  int StaffNo;
  int BranchNo;
  char Fname[15];
  char Lname[15];
  struct date DateOfBirth;
  float Salary;
  struct STAFF *next; // pointer to next record
};
```

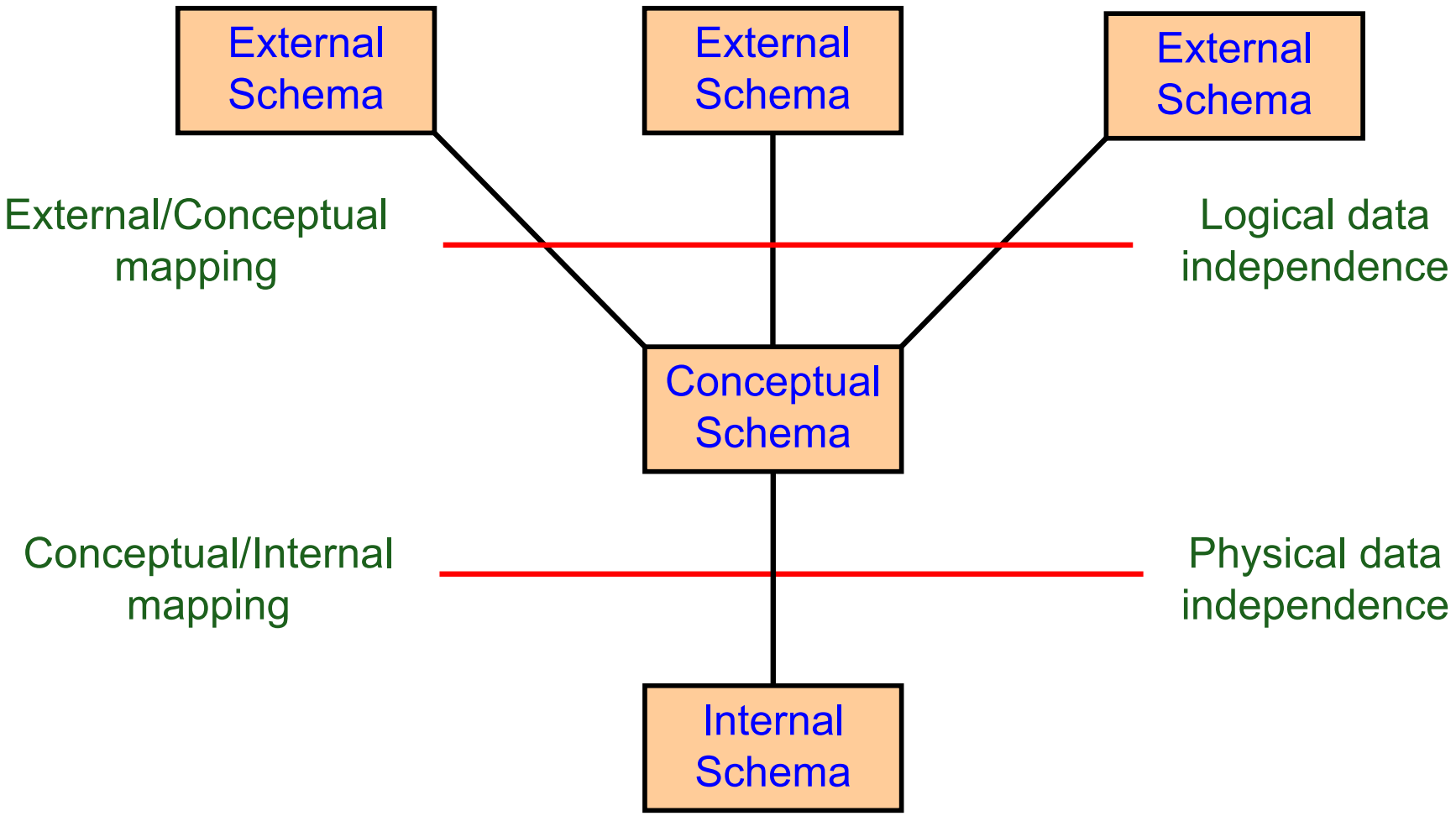
Notes on the Example

- The two external views are based on the conceptual view.
  - The *Age* field is derived from the *DOB* (Date of Birth) field.
  - The *Sno* field is mapped onto the *StaffNo* field of the conceptual record.
- The conceptual level is mapped onto the internal level.
- The internal level contains a physical description of the structure for the conceptual record expressed in a high-level language.
- Note that the order of the fields in the physical structure is different from that of the conceptual record.
- The physical structure contains a “pointer”, next. This will be simply the memory address at which the next record is stored. Thus the set of staff records may be physically linked together to form a chain.

Data Independence -I

- A major objective of the ANSI-SPARC architecture is to provide data independence meaning that upper levels are isolated from changes to lower levels.
- There are two kinds of data independence:
- *Logical data independence* refers to the immunity of external schemas to changes in the conceptual schema.
  - Changes to the conceptual schema (adding/removing entities, attributes, or relationships) should be possible without having to change existing external schemas or rewrite application programs.
- *Physical data independence* refers to the immunity of the conceptual schema to changes in the internal schema.
  - Changes to the internal schema (using different storage structures or file organisations) should be possible without having to change the conceptual or external schemas.

Data Independence -II



Database Languages

- A DBMS typically provides a *data sub-language* with which the database and its various schemas can be manipulated.
- Data sub-languages consist of two parts:
  - *Data Definition Language (DDL)*
  - *Data Manipulation Language (DML)*
- The DDL is used to specify the database schema and the DML is used to both update the database and extract information from it.
- They are called *sub-languages* because they do not include all of the facilities that one might expect of a high-level language.
  - i.e. There are no loops or conditional statements, etc.
- Thus, many systems allow the sub-language to be *embedded* in a high-level language like COBOL, Fortran, Pascal, Ada, or C.
- Most sub-languages also provide *interactive commands* that can be entered directly at a terminal and do not require embedding.

The Data Definition Language

- The DDL is a *descriptive language* that allows the user to describe and name the entities required and the relationships that may exist between the different entities.
- The database schema is specified by a set of definitions expressed in the DDL.
- The DDL may be used only to define a schema or modify an existing one. It cannot be used to manipulate data.
- Theoretically, there could be a different DDL for each type of database schema (external, conceptual, internal).
- However, in practice, the DBMS typically provides a single comprehensive DDL that allows specification of at least the external and conceptual schemas.

The Data Manipulation Language

- The DML is a language that provides a set of operations supporting the manipulation of the data held in the database.
- Data manipulation operations usually include:
  - *Inserting* new data or *Deleting* old data.
  - *Modifying* or *Retrieving* existing data.
- There are two types of DML which are distinguished by their underlying data retrieval constructs:
- With *procedural DMLs*, the programmer specifies what data is required and how to obtain it.
  - In this case, information retrieval is rather like writing a program.
- With *non-procedural DMLs*, the user specifies what data is to be retrieved in a single statement **without** specifying how the data should be obtained.
  - In this case, the DBMS is responsible for translating the DML statement into an optimal series of data manipulation operations.

Aside: Fourth Generation Languages (4GLs)

- There is no consensus about what constitutes a 4GL.
- For our purposes, 4GLs are generally non-procedural in nature.
- That is, they allow the user to specify what must be done without saying how it should be done.
  - This is very different from using a conventional language like Java (a 3GL) in which you have to specify how to do things.
  - With a 4GL, the *how* part is determined by the system.
- Examples include:
  - Form/Report generators
  - Application generators
  - SQL

Data Models - I

- A database schema is usually expressed using the DDL of a particular DBMS.
- However, this type of language is too low-level to describe the data requirements of an organisation in a readily understandable manner.
- People require a higher-level description of the schema that is organised using the concepts of a particular data model.
- A data model is an integrated collection of concepts for describing data, relationships between data, and constraints on the data in an organisation.

Data Models - II

- A data model comprises three components:
  - A structural part, consisting of a set of rules according to which databases can be constructed.
  - A manipulative part, defining the types of operations that are allowed on the data.
  - Possibly a set of integrity rules, which ensure that the stored data is accurate.
- Many data models have been proposed over the years.
  - Some are used to describe data at the external and conceptual levels, while others describe data at the internal level.
  - Some have greater success than others in hiding from end-users the underlying details of the physical storage of data.

Data Models - III

- Examples:
  - Network Model
  - Hierarchical Model
  - Relational Model
    - **(Entity-Relationship Model)**
  - Object-Oriented Model
  - Object-Relational Model
- The first two are older than the others, and by far the majority of database systems these days are based on the relational model.
- The last two are more modern, and incorporate the object-oriented approach to data representation.
- We have been concentrating on the relational model.

End of Lecture

Would you like to ask anything?

Don’t forget to read the notes again.