Bahria University-Karachi Campus

Software Design & Architecture

Lecture 5 of 16

Engr. Majid Kaleem

WEEKLY AGENDA

TENTATIVE WEEKLY DATES	TENTATIVE TOPICS
1	INTRODUCTION TO THE COURSE; DEFINING SOFTWARE ARCHITECTURE & DESIGN CONCEPTS
2	DESIGN PRINCIPLES; OBJECT-ORIENTED DESIGN WITH UML
3	SYSTEM DESIGN & SOFTWARE ARCHITECTURE; OBJECT DESIGN, MAPPING DESIGN TO CODE
4	FUNCTIONAL DESIGN; UI DESIGN; WEB APPLICATIONS DESIGN **ASSIGNMENT & QUIZ #1**
5	**MOBILE APPLICATION DESIGN; PERSISTENCE LAYER DESIGN**
6	CREATIONAL DESIGN PATTERNS
7	STRUCTURAL DESIGN PATTERNS **ASSIGNMENT & QUIZ #2**
8	BEHAVIORAL DESIGN PATTERNS
← MID TERM EXAMINATIONS →	
9	INTERACTIVE SYSTEMS WITH MVC ARCHITECTURE; SOFTWARE REUSE
10	ARCHITECTURAL DESIGN ISSUES; ARCHITECTURE DESCRIPTION LANGUAGES (ADLS)
11	ARCHITECTURAL STYLES/PATTERNS & DESIGN QUALITIES
12	ARCHITECTURAL STYLES/PATTERNS & DESIGN QUALITIES **ASSIGNMENT & QUIZ #3**
13	QUALITY TACTICS; ARCHITECTURE DOCUMENTATION
14	ARCHITECTURAL EVALUATION TECHNIQUES
15	MODEL DRIVEN DEVELOPMENT **ASSIGNMENT (PRESENTATIONS) & QUIZ #4**
16	REVISION WEEK
← FINAL TERM EXAMINATIONS →	

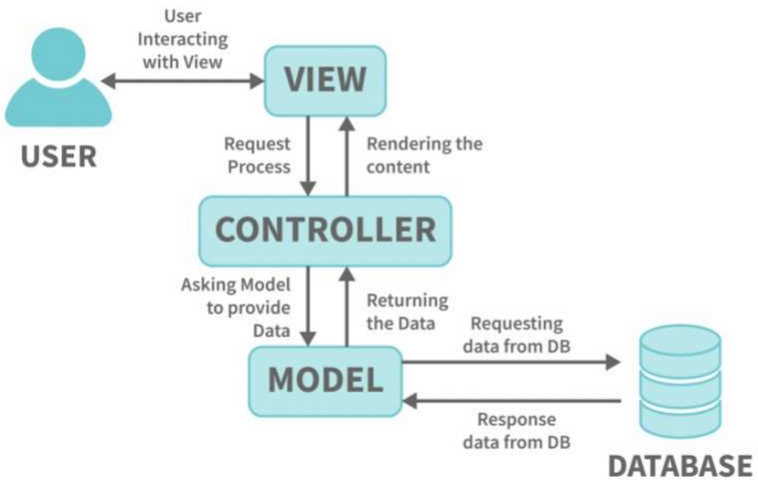
MOBILE APPLICATION DESIGN & ARCHITECTURE

- A dedicated course by the name: *"Software Applications for Mobile Devices"* is available in your roadmap to acquire in-depth knowledge on this topic.

MOBILE APPLICATION DESIGN & ARCHITECTURE

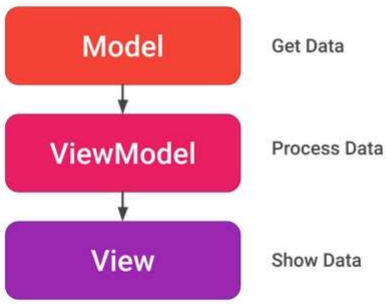
- The architecture of mobile applications generally follows the principles of software architecture, but with specific considerations for the unique characteristics of mobile devices.
- The most common architecture for mobile applications is the Model-View-Controller (MVC) pattern, which separates the application into three components:
  1. Model, which represents the data and business logic;
  2. View, which displays the user interface;
  3. Controller, which handles user input and updates the model and view accordingly.

MOBILE APPLICATION DESIGN & ARCHITECTURE



MOBILE APPLICATION DESIGN & ARCHITECTURE

- Another popular architecture for mobile applications is the *Model-View-ViewModel* (MVVM) pattern, which is similar to MVC but places more emphasis on data binding and separation of concerns.

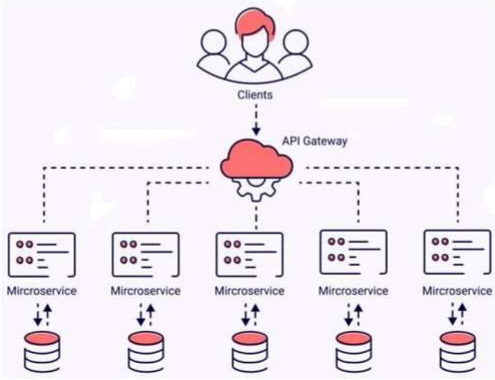


MOBILE APPLICATION DESIGN & ARCHITECTURE

- MVVM stands for Model-View-ViewModel. Each component has a different role.
  1. In the MVVM pattern, the Model represents the data and business logic of the application, just like in the MVC pattern.
  2. The View represents the user interface of the application, also like in the MVC pattern.
  3. However, the *ViewModel* replaces the *Controller* in the MVVM pattern.
  4. This is sometimes known as the logic layer as this is where you want to be processing all your data from the model before passing it to the view.
  5. The ViewModel is responsible for managing the data and logic of the View, and for mediating the communication between the View and the Model.

MOBILE APPLICATION DESIGN & ARCHITECTURE

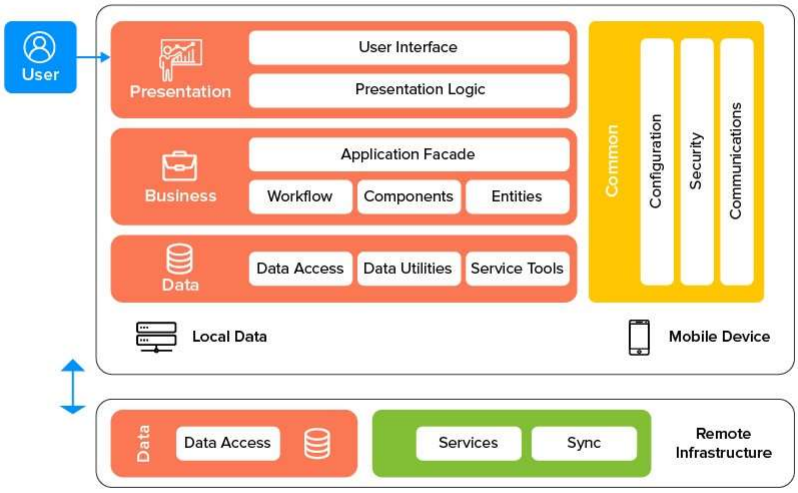
- Other architectures that are sometimes used for mobile applications include the Layered Architecture pattern, which separates the application into layers based on functionality; and the *Microservices* Architecture pattern, which breaks the application into small, independent services that communicate with each other.



MOBILE APPLICATION DESIGN & ARCHITECTURE

- Mobile apps are typically structured using multilayered architectures, including a user interface layer, a business layer, and a data layer.
- With mobile apps you have the choice of building a thin Web-based client or a rich client.
- With a *thin client*, only the user interface resides on the mobile device, whereas the business and data layers reside on a server.
- With a *rich client* all three layers may reside on the mobile device itself.

MOBILE APPLICATION DESIGN & ARCHITECTURE

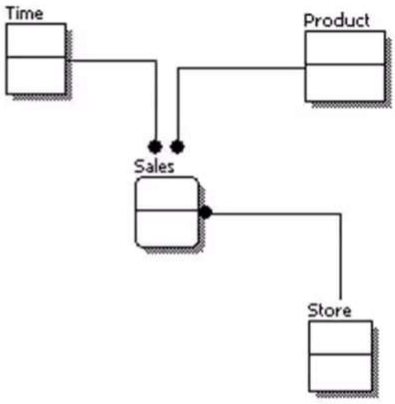


PERSISTENCE LAYER DESIGN

- A dedicated course by the name: *"Relational Database Management Systems"* is available in your roadmap to acquire in-depth knowledge of persistence layer.
- Persistence layer design means designing of *database* or data store layer.
- Several styles are available as described on the following slides.

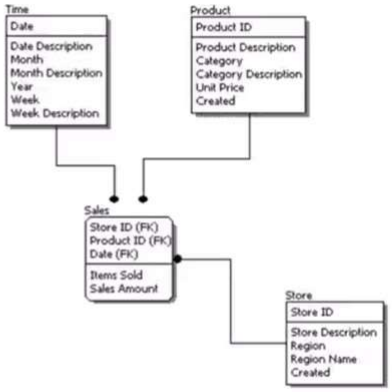
WHAT IS A DATA MODEL

1. Conceptual Data Model
   - Highly abstract
   - Easily understood
   - Easily enhanced
   - Only "Entities" visible
   - Abstract relationships
   - No software tool is required to define it



WHAT IS A DATA MODEL

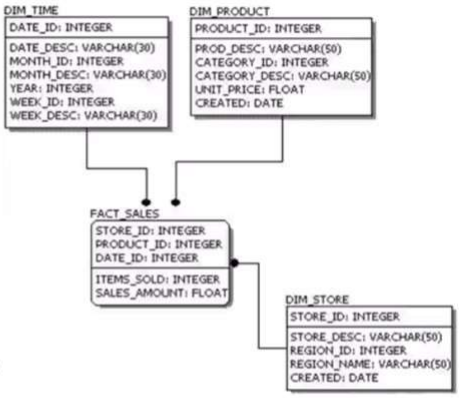
2. Logical Data Model
   - Presence of Attributes for each Entity
   - Key Attributes
   - Non-Key Attributes
   - Primary Key-Foreign Key Relationships



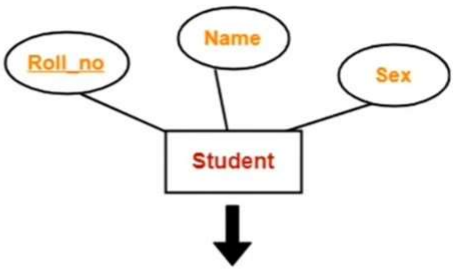
- Presence of Attributes for each Entity
- Key Attributes
- Non-Key Attributes
- Primary Key – Foreign Key Relationships
- User Friendly Attribute names
- More detailed than Conceptual Model
- Database agnostic
- Bit more effort required to enhance, in comparison to Conceptual Model
- Data Modeling tools like ERWin or PowerDesigner can be used to create Logical Data Models. This can be automatically converted to a Physical Data Model with the help of these tools.

WHAT IS A DATA MODEL

3. Physical Data Model
   - Presence of Attributes for each Entity
   - Key Attributes
   - Non-Key Attributes
   - Primary Key-Foreign Key Relationships

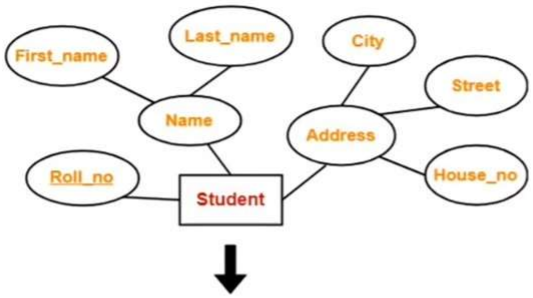


- Entities referred to as Tables
- Attributes referred to as Columns
- Database compatible table names
- Database compatible column names
- Database specific data types
- Difficult for users to understand
- Significantly more effort required to enhance in comparison to Logical Model
- Will include indexes, constraints, triggers & other DB objects
- Difficult to port to a different database, once design is finalized.
- Tools like ERWin and PowerDesigner can help in automatically porting over the Logical Data Model to Physical Data Models of different versions.



Roll_no	Name	Sex

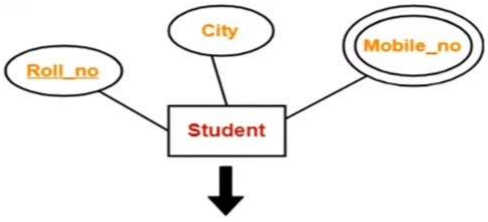
Schema : Student (Roll_no , Name , Sex)



Roll_no	First_name	Last_name	House_no	Street	City

Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

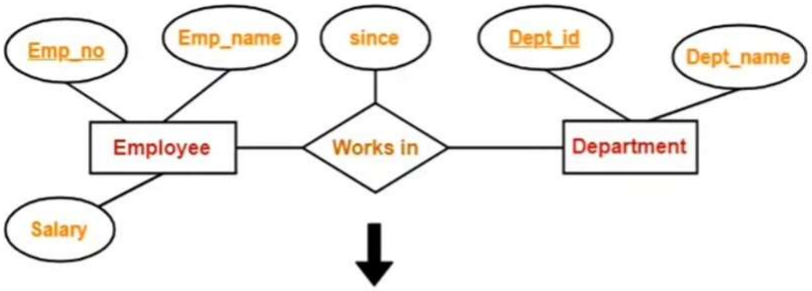
26-Mar-2023 Engr. Majid Kaleem 17



Roll_no	City

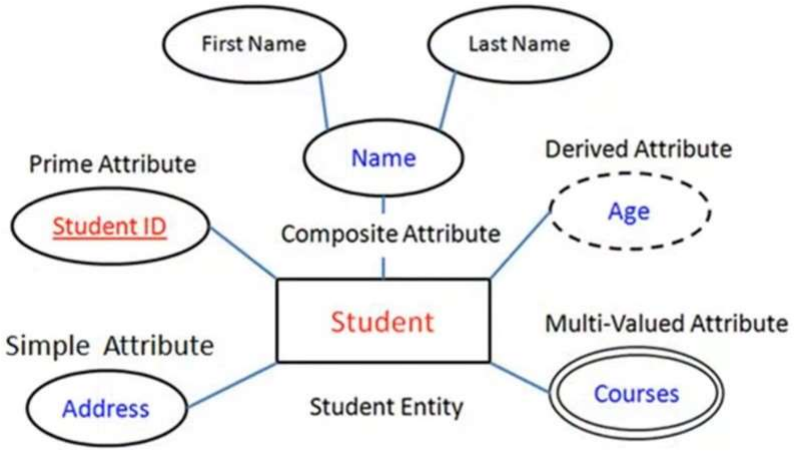
Roll_no	Mobile_no

26-Mar-2023 Engr. Majid Kaleem 18

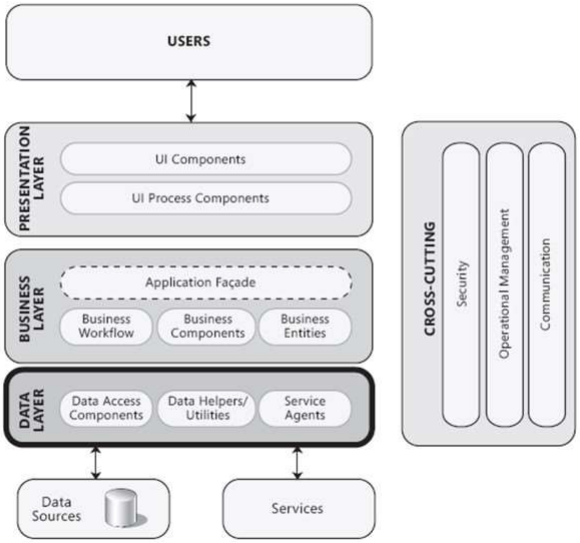


Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)



PERSISTENCE LAYER TYPICAL ARCHITECTURE

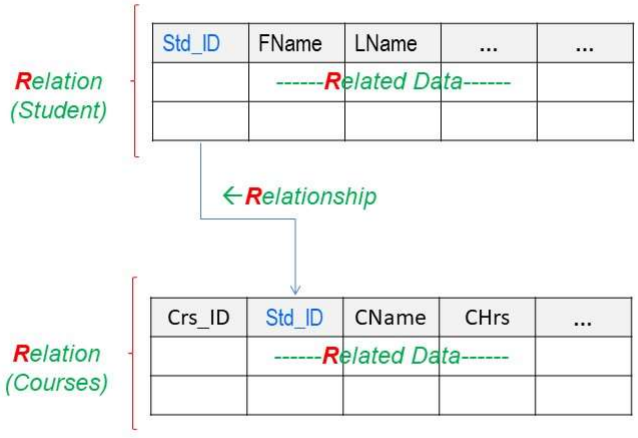


A typical application showing the data layer and the components it may contain

21

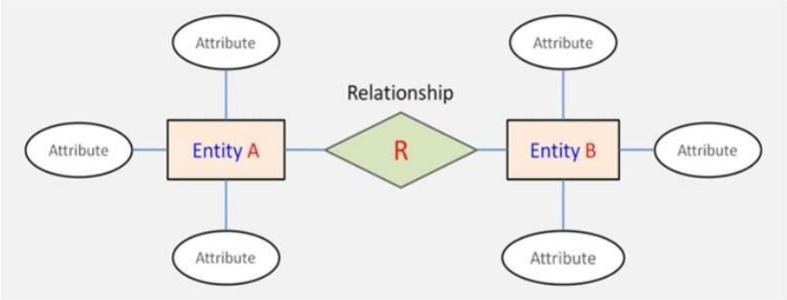
RELATIONAL DATABASE

Relation=Table



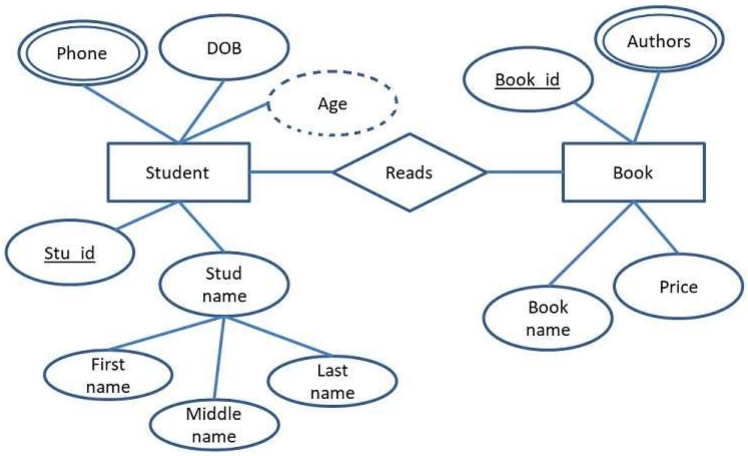
RELATIONAL DATABASE EXAMPLE

- Entity-Relationship Diagram (ERD) - Pictorial representation of a database model.



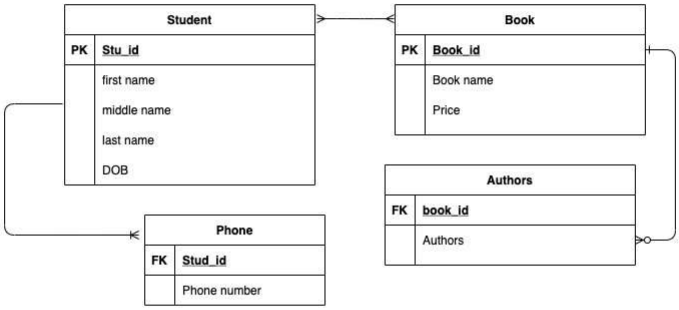
RELATIONAL DATABASE EXAMPLE

- ER Model:



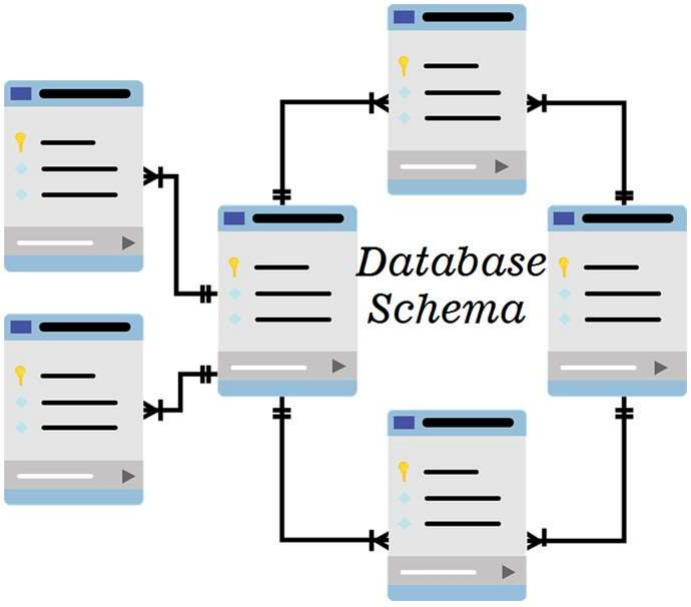
RELATIONAL DATABASE EXAMPLE

- Relational Model:



DATABASE SCHEMA EXAMPLE

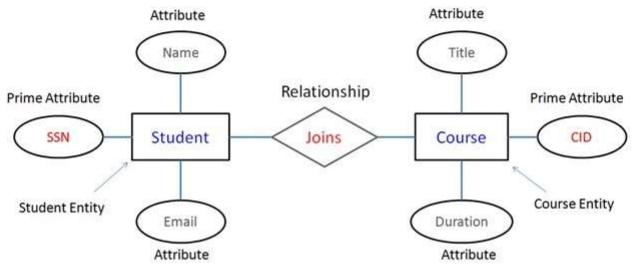
- X



- Identifying entities is the first step in Data Modelling.
- **How to identify an entity from a given problem**
  - Search for *nouns*, like Teacher, Doctor, etc.
  - Classify nouns to get a wider picture of the entities.
  - Read the problem description repeatedly.
  - Entities are like Persons, Students, Teachers, and Courses.

RELATIONAL DATABASE EXAMPLE

- Converting an ER model into Relational model:

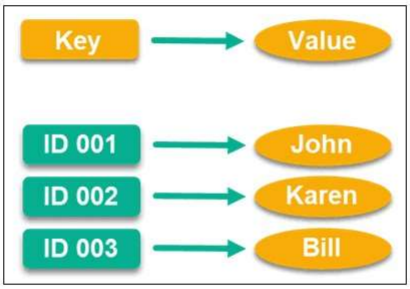


SSN	Name	Email

CID	Title	Duration

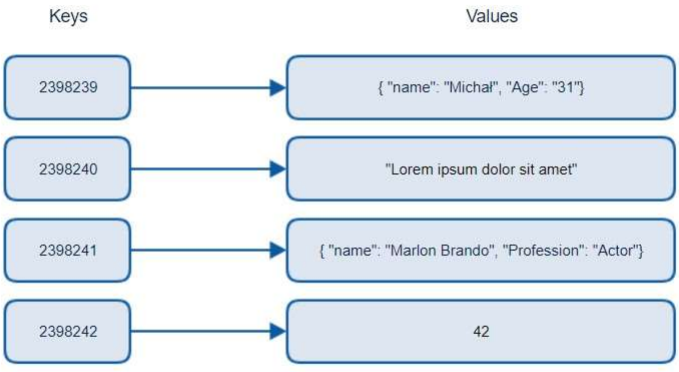
KEY/VALUE DATABASE EXAMPLE

- A key/value store implements arguably the simplest of the NoSQL storage mechanisms as shown below:



29

KEY/VALUE DATABASE EXAMPLE

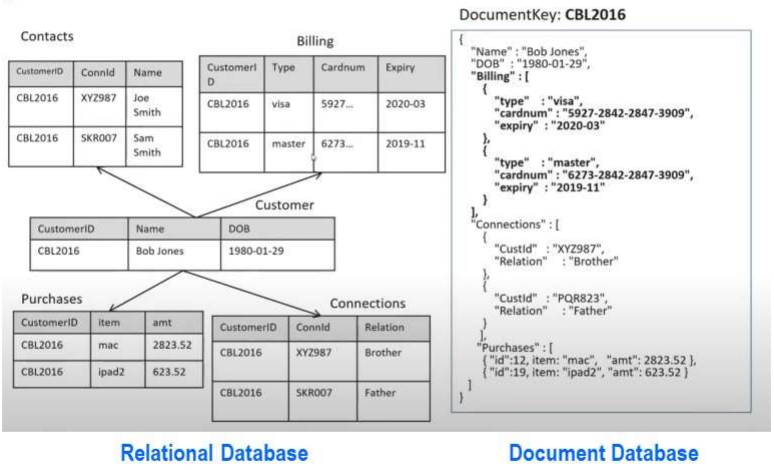


DOCUMENT DATABASE

- A *document database* is similar in concept to a key/value store except that the values stored are documents.
- A document is a collection of named fields and values, each of which could be simple scalar items or compound elements such as lists and child documents.
- The fields in the documents are exposed to the database management system, enabling an application to query and filter data by using the values in these fields.

31

DOCUMENT DATABASE EXAMPLE



DOCUMENT DATABASE EXAMPLE

Relational

ID	first_name	last_name	cell	city	year_of_birth	location_x	location_y
1	'Mary'	'Jones'	'516-555-2048'	'Long Island'	1986	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

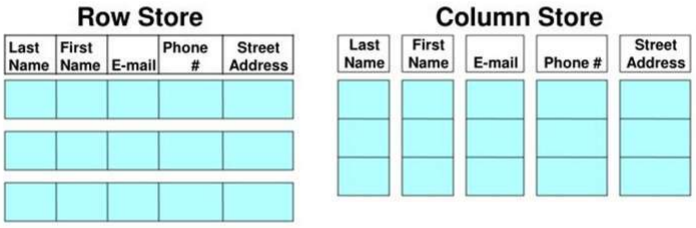
ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

MongoDB

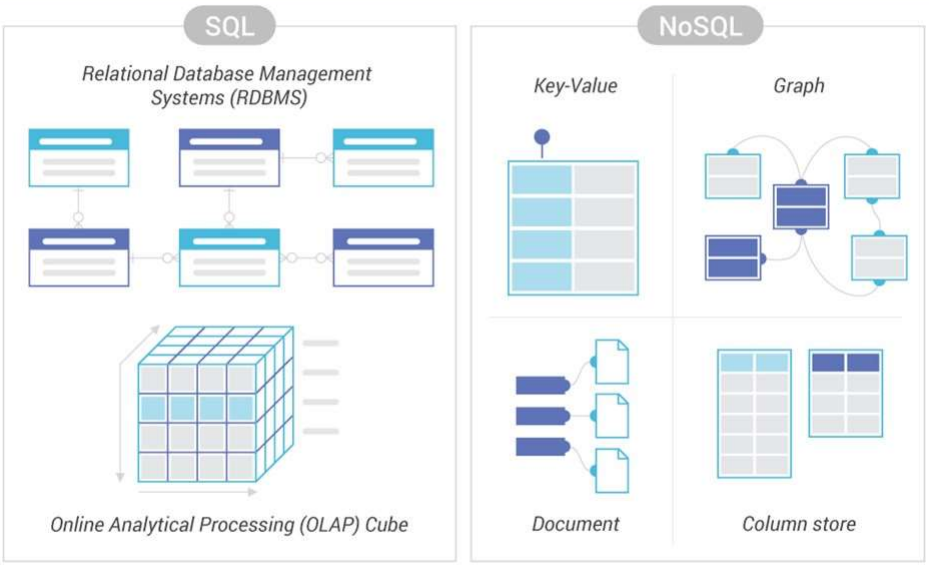
```
first_name: "Mary",
last_name: "Jones",
cell: "516-555-2048",
city: "Long Island",
year_of_birth: 1986,
location: {
        type: "Point",
        coordinates: [-73.9876, 40.7574]
},
profession: ["Developer", "Engineer"],
apps: [
 { name: "MyApp",
   version: 1.0.4 },
 { name: "DocFinder",
   version: 2.5.7 }
],
cars: [
   { make: "Bentley",
     year: 1973 },
   { make: "Rolls Royce",
     year: 1965 }
 ]
```

Relational Database **Document Database**

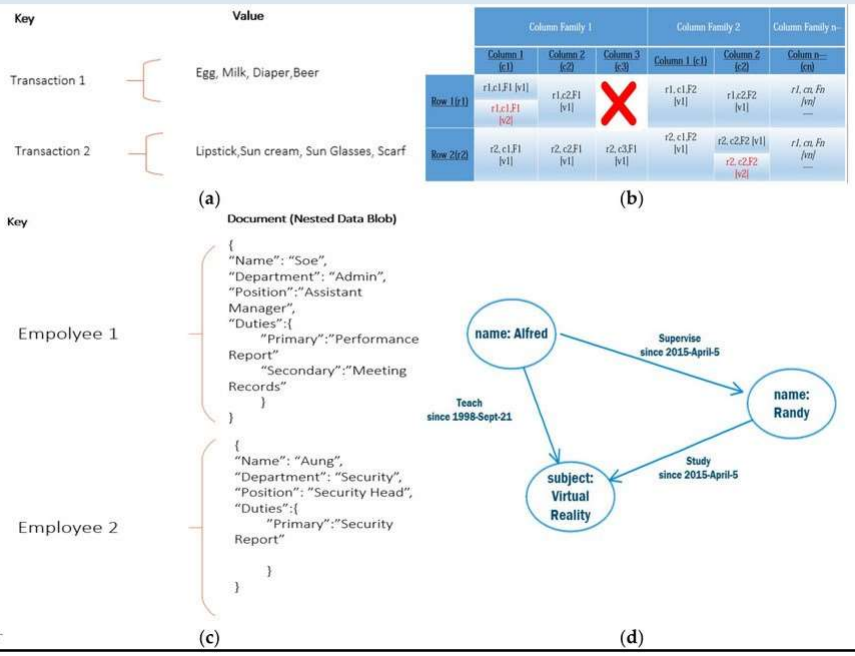
WIDE-COLUMN DATABASE EXAMPLE



PERSISTENCE LAYER DESIGN-SUMMARY



PERSISTENCE LAYER DESIGN-SUMMARY



PERSISTENCE LAYER DESIGN-SUMMARY

1. **Relational database** — Represents the database as a collection of relations. A relation is nothing but a table (rows & columns) of values.
2. **Key-value store** — Stores data with simple indexed keys and values. Examples include Oracle NoSQL database, Redis, Aerospike, Oracle Berkeley DB, Voldemort, Amazon DynamoDB and Infinity DB.
3. **Document database** — A more complex and structured version of the key-value model, which gives each document its own retrieval key. Examples include Orient DB, MarkLogic, MongoDB, IBM Cloudant, Couchbase, and Apache CouchDB.
4. **Wide column store** — Uses tables, rows and columns. But the format and naming of the columns can vary in different rows within the same table. Examples include Apache Cassandra, Scylla, Datastax Enterprise, Apache HBase, Apache Kudu, Apache Parquet and MonetDB.
5. **Graph database** — Presents interconnected data as a logical graph. Examples include Neo4j, JanusGraph, FlockDB and GraphDB.

VARIOUS NAMES FOR SAME CONCEPTS

Theory	Database	File	SOM	ER
Relation	Table	File	Class	Entity Set
Tuple	Row	Record	Object	Entity
Attribute	Column	Field	Attribute	Attribute

```
If(anyQuestions)
{
  askNow();
}
else
{
  thankYou();
  submitAttendance();
  endClass();
}
```


REFERENCES

1. ***Software Architecture***, *Perspectives on an Emerging Discipline By Mary Shaw & David Garlan*
2. ***The Art of Software Architecture***, *Design Methods & Techniques By Stephen T. Albin*
3. ***Essential Software Architecture***, *By Ian Gorton*
4. ***Microsoft Application Architecture Guide***, *By Microsoft*
5. ***Design Patterns***, *Elements of Reusable Object-Oriented Software By by Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides*
6. ***Refactoring, Improving the Design of Existing Code***, *By Martin Fowler & Kent Beck*