Introduction Database System

Contents

- Levels of Architecture in a DBMS
- Client/Server Architecture
- Web Application Architechture – Client / Server Architecture
- Distributed processing

1- Levels of Architecture in a DBMS

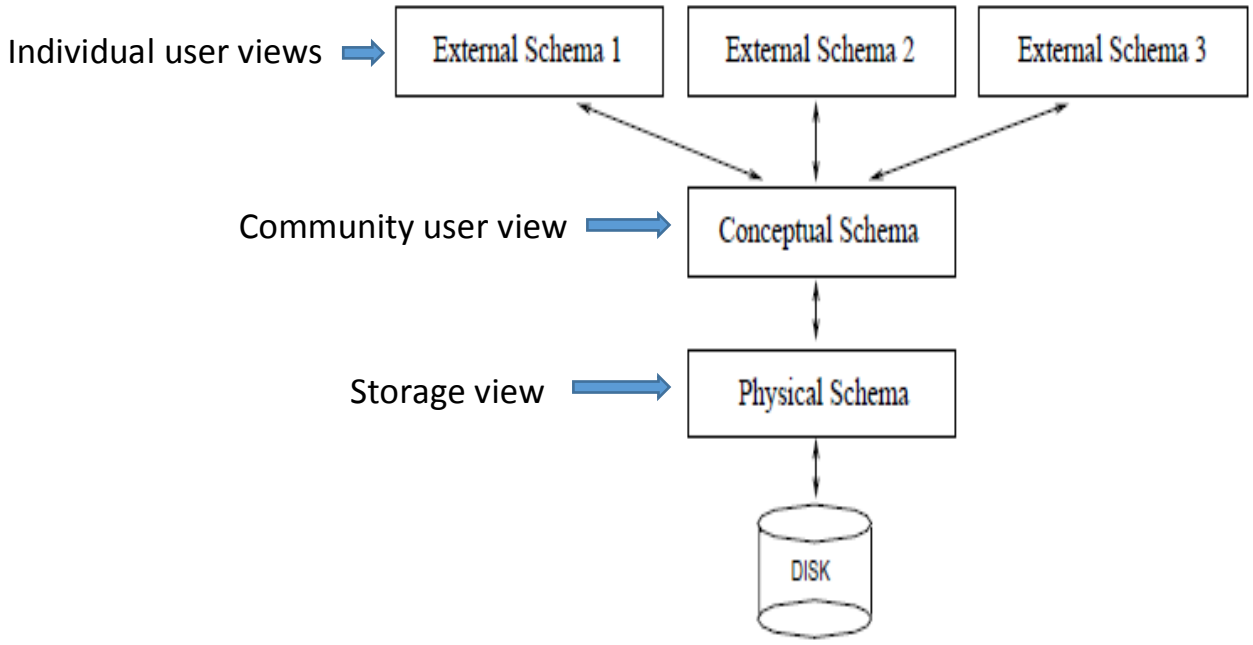


Figure 2.1 Levels of Architecture in a DBMS

The data in a DBMS is described at three levels of architecture, as illustrated in Figure 2. The database description consists of a schema at each of these three levels of architecture: the ***conceptual***, ***physical***, and ***external schemas***.

A ***data definition language*** (***DDL***) is used to define the external and conceptual schemas. All DBMS vendors also support SQL commands to describe aspects of the physical schema.

☒ ***Physical Schema***

The physical schema specifies additional storage details. Essentially, the physical schema summarizes how the relations described in the conceptual schema are actually stored on secondary storage devices such as disks and tapes. We must decide what file organizations to use to store the relations, and create auxiliary data structures called indexes to speed up data retrieval operations. A sample physical schema for the university database follows:

- ✓ Store all relations as unsorted files of records. (A file in a DBMS is either a collection of records or a collection of pages, rather than a string of characters as in an operating system.)
- ✓ Create indexes on the first column of the Students, Faculty, and Courses relations, the *sal* column of Faculty, and the capacity column of Rooms.

Decisions about the physical schema are based on an understanding of how the data is typically accessed. The process of arriving at a good physical schema is ***called physical database design***.

☒ ***conceptual Schema***

The schema (sometimes called the logical schema) describes the stored data in terms of the data model of the DBMS. In a relational DBMS, the conceptual schema describes all relations that are stored in the database. In our sample university database, these relations contain information about entities, such as students and faculty, and about relationships, such as students' enrollment in courses. All student entities can be described using records in a Students relation, as we saw earlier. In fact, each collection of entities and each collection of relationships can be described as a relation, leading to the following conceptual schema:

Students (sid:string, name: string, login: string, age: integer, gpa: real) Faculty(_d: string, fname: string, sal: real) Courses(cid: string, cname: string, credits: integer) Rooms(rno: integer, address: string, capacity: integer) Enrolled(sid: string, cid: string, grade: string) Teaches(_d: string, cid: string) Meets In(cid: string, rno: integer, time: string)

The choice of relation ,and the choice of fields for each relation, is not always obvious, and the process of arriving at a good conceptual schema is called ***conceptual database design***.

☒ ***External Schema***

External schemas, which usually are also in terms of the data model of the DBMS, allow data access to be customized (and authorized) at the level of individual users or groups of users. Any given database has exactly one conceptual schema and one physical schema because it has just one set of stored relations, but it may have several external schemas, each tailored to a particular group of users. Each external schema consists of a collection of one or more views and relations from the conceptual schema. A view is conceptually a relation, but the records in a view are not stored in the DBMS. Rather, they are computed using a definition for the view, in terms of relations stored in the DBMS. The external schema design is guided by end user requirements.

For example, we might want to allow students to find out the names of faculty members teaching courses, as well as course enrollments. This can be done by defining the following view:

Courseinfo(cid: string, fname: string, enrollment: integer)

A user can treat a view just like a relation and ask questions about the records in the view. Even though the records in the view are not stored explicitly, they are computed as needed. We did not include Courseinfo in the conceptual schema because we can compute Courseinfo from the relations in the conceptual schema, and to store it in addition would be redundant. Such redundancy, in addition to the wasted space, could lead to inconsistencies. For example, a tuple may be inserted into the Enrolled relation, indicating that a particular student has enrolled in some course, without incrementing the value in the enrollment field of the corresponding record of Courseinfo (if the latter also is part of the conceptual schema and its tuples are stored in the DBMS).

☒ ***Data Independence***

A very important advantage of using a DBMS is that it offers data independence. That is, application programs are insulated from changes in the way the data is structured and stored. Data independence is achieved through use of the three levels of data abstraction; in particular, the conceptual schema and the external schema provide distinct benefits in this area. Relations in the external schema (view relations) are in principle generated on demand from the relations corresponding to the conceptual schema. If the underlying data is reorganized, that is, the conceptual schema is changed, the definition of a view relation can be modified so that the same relation is computed as before. For example, suppose that the Faculty relation in our university database is replaced by the following two relations:

Faculty public(fid: string, fname: string, office: integer)
Faculty private(fid: string, sal: real)

2- Client/Server Architecture

Today client/server computing is a fact of life. The Internet—and its intranet and extranet derivatives is perhaps the most pervasive example of client (also is called frontend)/server (also called backend) computing, and it has taken center stage with regard to application development. Because of the Internet's wide reach and acceptance, you should know what client/server computing is; what its components are; how the components interact; and what effects client/server computing has on review database design, implementation, and management. The figure (1.1) is show the Client/Server Systems

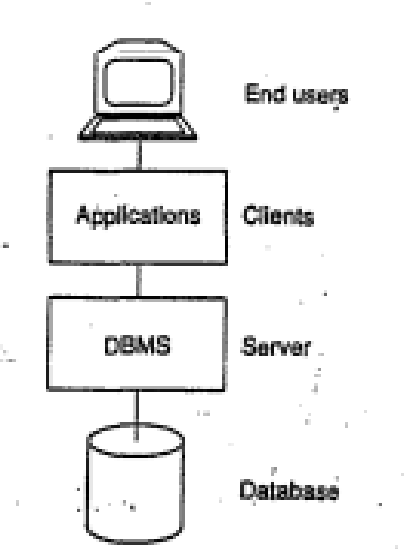


Fig. 2.5 Client/server architecture

- ***Client***

  A client is a single-user workstation that provides presentation services and the appropriate computing, connectivity and the database services and the interfaces relevant to the business need.

- ***Server***

  A server is one or more multi-user processors with share memory providing computing, connectivity and the database services and the interfaces relevant to the business need.

  The server is jest the DBMS itself. It supports all the basic DBMS function, the major function of DBMS is(refer of figure 2.1)

- ✓ ***Data definition***

  The DBMS must be able to concept data definition(external schemas , the conceptual schema , the internal schema , and all associated mapping)

  تعريف البيانات :

  يجب أن يكون قادرا على تعريف مفهوم البيانات (مخططات خارجية، مخطط مفاهيمي، مخطط داخلي، وجميع الخرائط المرتبطة بها) نظم إدارة قواعد البيانات

- ✓ ***Data manipulation***

  The DBMS must be able to handle requests to retrieve, update, or delete existing data in the data or to add new data to the database.

  التلاعب البيانات :

  يجب أن تكون قادرة على التعامل مع طلبات لاسترداد أو تحديث أو حذف البيانات الموجودة في DBMS. البيانات أو إضافة بيانات جديدة إلى قاعدة البيانات

- ✓ ***In general DML request is can be "planned" or "unplanned"***

  - **A planned**

    Request is one for which the need was foreseen well in advance of the time at which the request is executed. The DBA will probably have tuned the physical database design in such a way as to good performance for planned requests.

  - **An unplanned**

    Request by contrast is an *ad hoc* query i.e. a request for which the need was not seen in advance, but instead arose in spur-of-the-moment fashion. The physical database design might or might not be ideally suited for the specific request under consideration.

    في طلب DML له "المخطط" أو "غير مخطط" العام ويمكن أن يكون :

    - والمخطط لها

    طلب واحد والتي كان من المتوقع على ضرورة أن يتم ذلك قبل الوقت الذي يتم تنفيذ الطلب. تصميم قاعدة البيانات البدني في مثل هذه الطريقة إلى الأداء الجيد DBA وربما قد ضبطها لطلبات المخطط لها.

    - ومن غير المخطط لها

    طلب على النقيض من ذلك هو استعلام مخصص أي طلب الذي لم ير حاجة مقدما، ولكن بدلا من ذلك نشأ في حفز من بين لحظة الموضة. قد تصميم قاعدة البيانات البدني أو قد لا تكون مناسبة بشكل مثالي للطلب محدد قيد النظر.

- ✓ ***Optimization and execution***

  DML requests, planned or unplanned, must be processed by the ***optimizer*** component, whose purpose is to determine an efficient way of implementing the request. The optimized

requests are then executed under the control of the ***run-time manager***.

الأمثل والتنفيذ

، مخططة أو غير مخططة، يجب أن تتم معالجتها بواسطة المكون محسن، الذي DML طلبات يهدف إلى تحديد وسيلة فعالة لتنفيذ الطلب. ثم يتم تنفيذ طلبات الأمثل تحت سيطرة مدير وقت التشغيل.

- ✓ ***Data security and integrity***

The DBMS must monitor user requests and reject any attempts to violate the security and integrity constraints defined by the DBA.

من البيانات وسلامة

نظم إدارة قواعد البيانات يجب مراقبة طلبات المستخدمين ورفض أي محاولات لانتهاك القيود الأمنية والنزاهة التي حددها DBA.

- ✓ ***Data recovery and concurrency***

The DBMS more likely, some other related software component, usually called the ***transaction manager*** or ***transaction-processing monitor*** (TP monitor) must enforce certain recovery and concurrency controls.

ستعادة البيانات والتزامن

نظم إدارة قواعد البيانات على الأرجح، بعض مكونات البرامج الأخرى ذات الصلة، وعادة ما يجب فرض ضوابط معينة (TP رصد) تسمى إدارة المعاملات أو شاشة معالجة المعاملات الانتعاش والتزامن.

- ✓ ***Data dictionary***

The DBMS must provide a ***data dictionary*** function. The data dictionary can be regarded as a database in its own right (but a system database rather than a user database). The dictionary contains "data about the data "(sometimes-called ***metadata*** or ***descriptors***. In particular, all of the various schemas and mappings (external, conceptual, etc.), and all of the various security and integrity constraints will be stored, in both source and object form, in the dictionary. A comprehensive dictionary will also include much additional information, showing, for instance, which programs use which parts of the database, which users require which reports, and so on.

قاموس البيانات

توفير وظيفة قاموس البيانات. ويمكن اعتبار أن قاموس البيانات قاعدة DBMS يجب على بيانات في حد ذاتها (ولكن قاعدة بيانات النظام بدلا من قاعدة بيانات المستخدم). يحتوي القاموس "بيانات عن البيانات" (أحيانا يسمى الفوقية أو واصفات وعلى وجه الخصوص، سيتم تخزين جميع من مختلف المخططات وتعيينات (الخارجية والمفاهيمية، الخ)، وجميع من مختلف القيود الأمنية والنزاهة، في كل من المصدر وشكل الكائن، في القاموس. سوف يتضمن القاموس الشامل

أيضا معلومات إضافية كبيرة، والتي تبين، على سبيل المثال، والتي تستخدم البرامج التي أجزاء من قاعدة البيانات، والتي تتطلب المستخدمين التي تقدم تقاريرها، وهلم جرا.

✓ ***Performance***

It goes without saying that the DBMS should perform all of the tasks identified above as efficiently as possible.

 الأداء

يجب تنفيذ كافة المهام المحددة أعلاه بأكبر قدر من الكفاءة DBMS وغني عن القول أن.

.

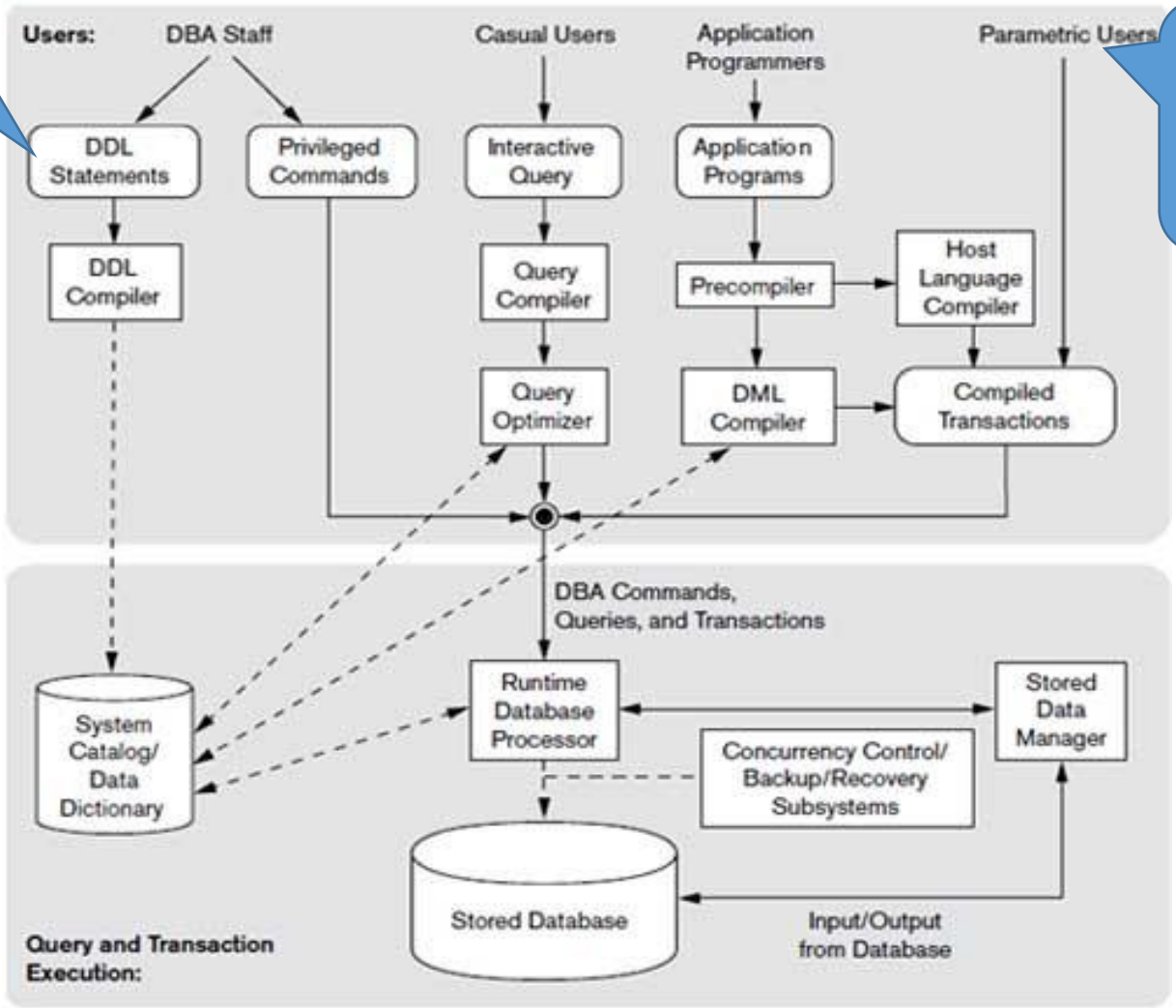


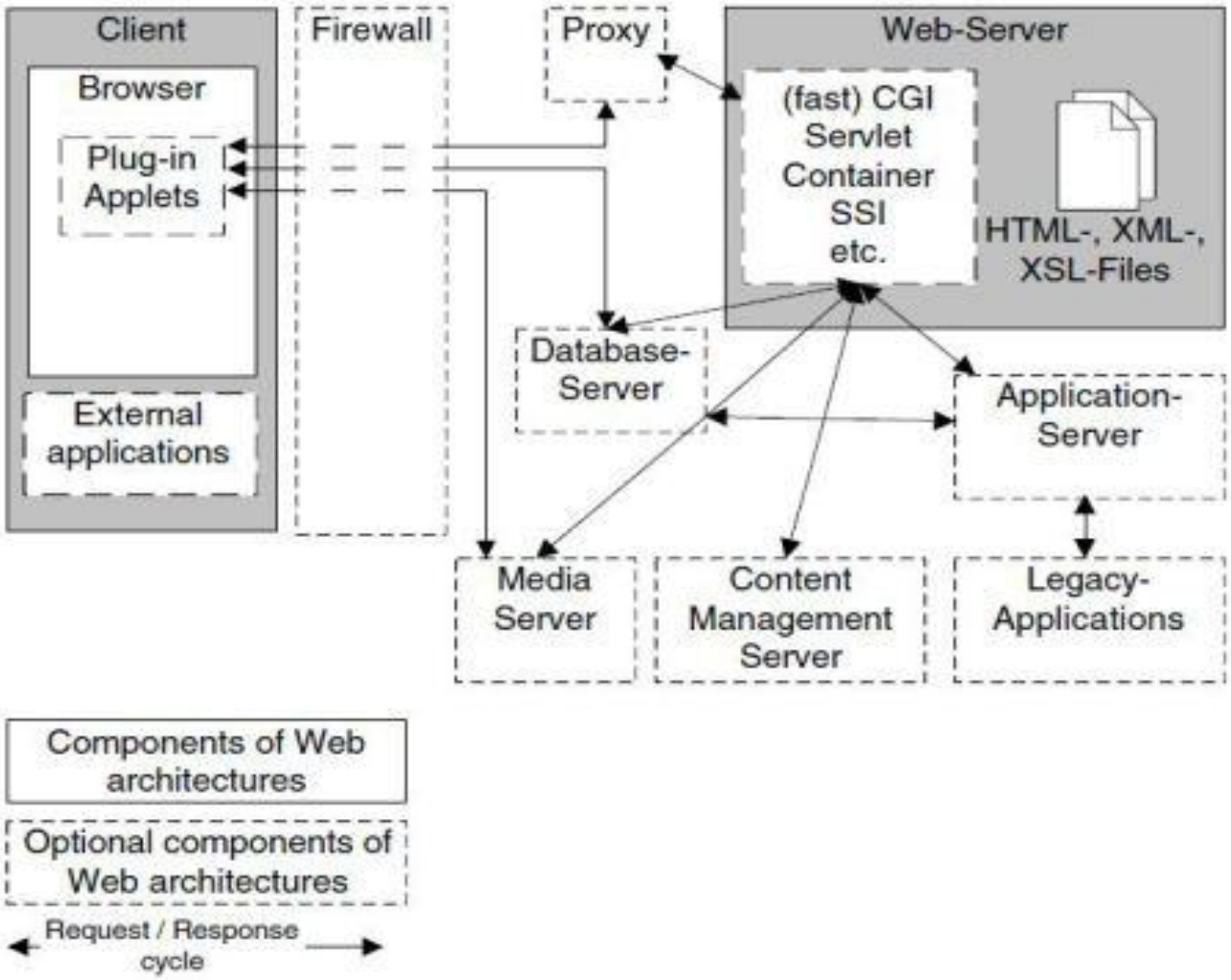
Figure 2-1 Major DBMS function and components

3-Web Application Architechture – Client / Server Architecture

Architecture is the structure of software systems which is divided into different perspectives, allowing us to manage the complexity of software systems in a better and easier way.

- ✓ ***framework*** is a reusable software system with general functionality already implemented. It can be specialized into a ready-to-use application.
- ✓ ***Benefits***: The simple reuse of architecture and functionality
- ✓ ***Drawbacks***: A high degree of training effort, a lack of standards for the integration of different frameworks, and the resulting dependence on manufacturers.

Components of a Generic Web Application Architecture



Web browser sends a request to Web server and the response to this request is sent back.

- ✓ ***Client*** = User agent. It is controlled by a user to operate the Web application. The client's functionality can be expanded by installing plug-ins, add-ons and applets.
- ✓ ***Firewall***: A software or hardware regulating the communication between insecure networks (e.g., the Internet) and secure networks (e.g., corporate LANs). This communication is filtered by access rules.
- ✓ ***Proxy***: A proxy is typically used to temporarily store Web pages in a cache. However, proxies can also assume other functionalities, e.g., adapting the contents for users (customization), or user tracking.
- ✓ ***Web server***: A Web server is a software that supports various Web protocols like HTTP, and HTTPS, etc., to process client requests.

Example

1. **Open source Apache Web Server**
2. **IIS Web Server**
3. **Tomcat Server**

☒ **Database server**: This server normally supplies an organization's production data in structured form.

Example

1. **Open source MySQL**
2. **MS SQL Server**
3. **Oracle database server**

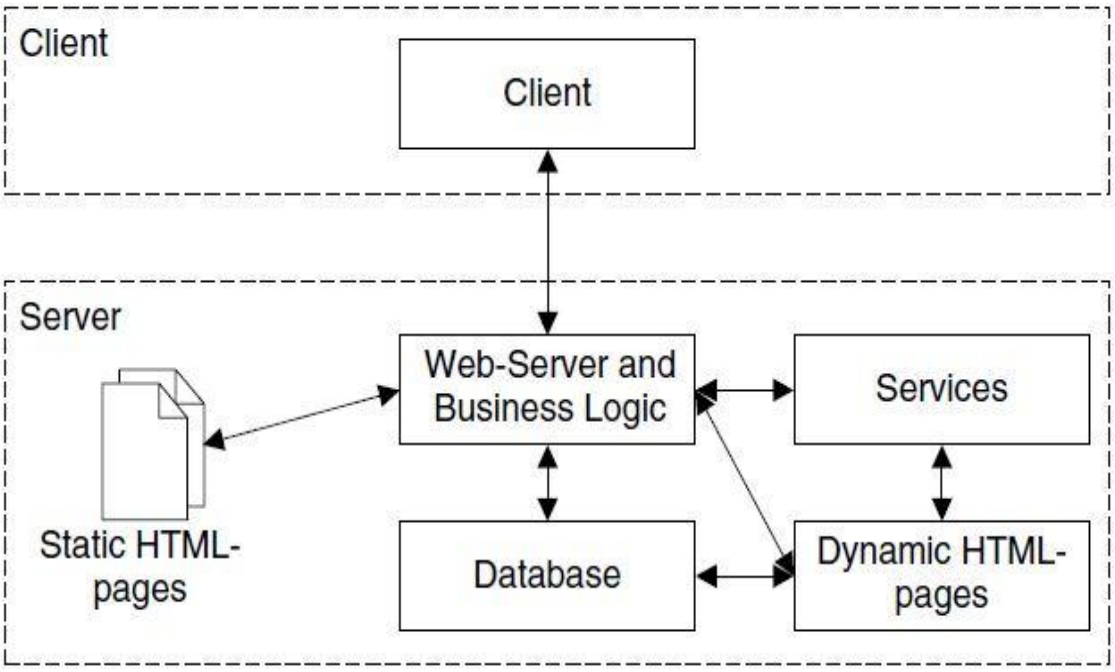
☒ ***Media server***: *This component is primarily used for content streaming of non-structured bulk data like audio and video.*

☒ ***Content management server:*** *Similar to a database server, a content management server holds contents to serve an application. These contents are normally available in the form of semi-structured data, e.g., XML documents.*

☒ ***Application server***: *An application server holds the functionality required by several applications, e.g., workflow or customization.*

☒ ***Legacy application***: A legacy application is an older system that should be integrated as an internal or external component.

☒ *-Layer Architectures = Client / Server Architechture*



It uses a Web server to provide services to a client.

A client request can point directly to static HTML pages, without requiring any processing logic on the server layer, or it can access a database via the application logic on the Web server (e.g., in the form of CGI scripts).

Dynamic HTML pages include script instructions directly in the HTML code, e.g., when SSI (Server-Side Include) is used, and they are interpreted either by databases with HTML functionality or by a Web server. The application logic, or dynamic HTML pages, can use services (e.g., user identification or data encryption) when the HTML response is generated.

This architecture is suitable particularly for simple Web applications. In contrast, a multilayer architectural approach is required for more demanding applications which are accessed by a large number of concurrent clients or which provide complex business processes requiring the access to legacy systems, amongst others.

4- Distributed processing

Distributed processing means that distinct machines can be connected together into a communication network such as the Internet. Such that a single data-processing task can span several machines in the network. Communication among the various machines is handled by some kind of network management software possibly an extension of Data Communication Manager (DC).

Many levels or varieties of distributed processing are possible. One simple case involves running the DBMS backend (the server) on one machine and the application frontends (the clients) on another. Refer to figure 2-2.

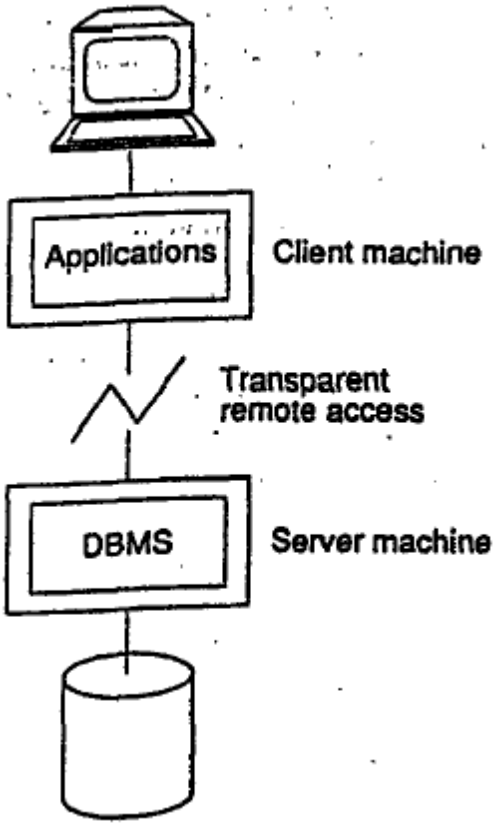


Figure 2-2 Cline(s) and server running on different machines

Figure 2-2 is which client and server run on different machines. Indeed, there are many arguments in favor of such a scheme:

- ✓ The first is basically just the usual parallel-processing argument: namely, many processing units are now being applied to the overall

task, and server (database) and client (application) processing are begin done in parallel.

- ✓ Furthermore, the server machine might be a custom-built machine that is tailored to the DBMS function (a "database machine") and might thus provide better DBMS performance.
- ✓ Likewise, the client machine might be a personal workstation, tailored to the needs of the end user and thus able to provide better interface, high availability, faster responses and overall improved ease of use to the user.
- ✓ Several different client machine might be able (in fact, typically will be able) to access the same server machine. Thus a single database might be shared across several distinct client system(see Figure 2-3)

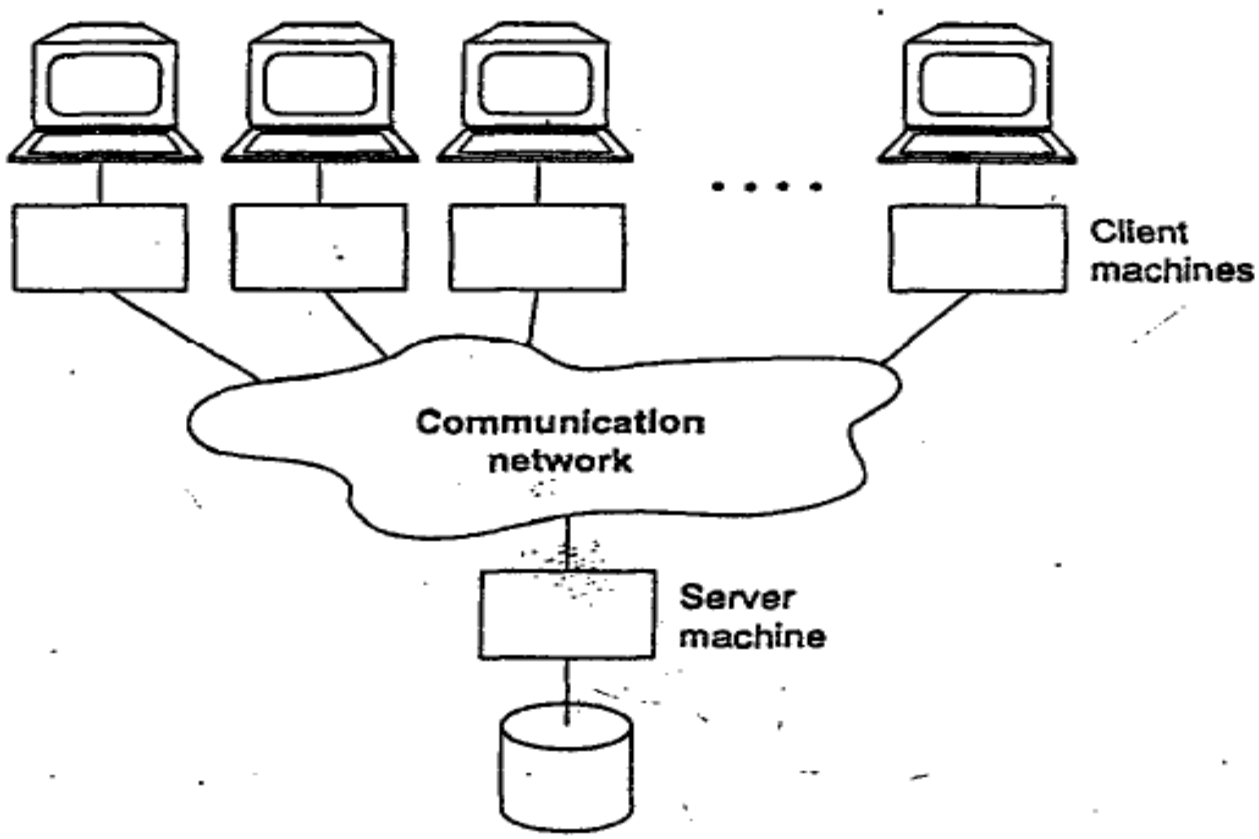


Figure 2-3 One server machine, many client machines

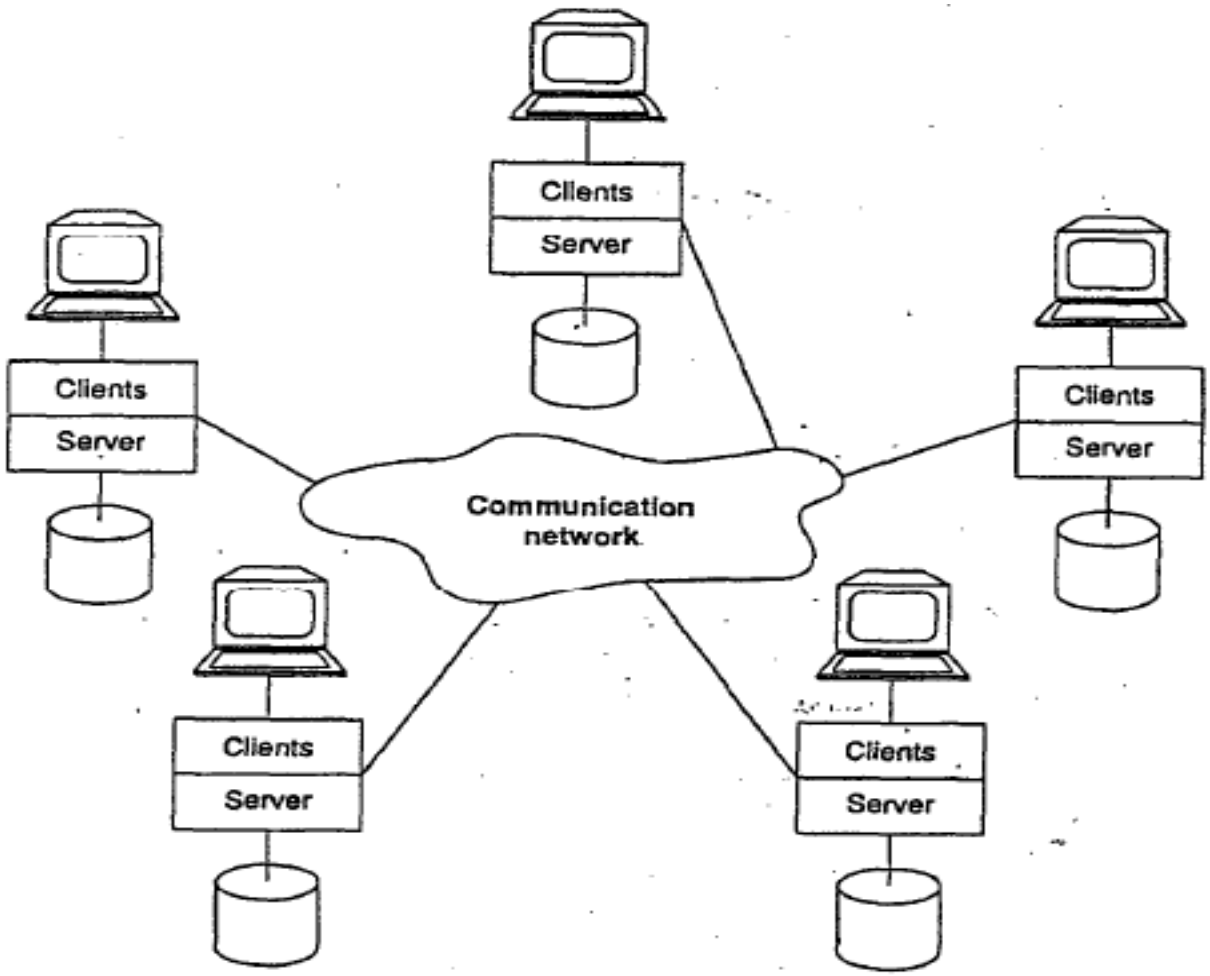


Figure 2-4 Each machine runs both client(s) and server

❖ DDL

Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- ✓ CREATE - to create objects in the database
- ✓ ALTER - alters the structure of the database
- ✓ DROP - delete objects from the database
- ✓ TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- ✓ COMMENT - add comments to the data dictionary
- ✓ RENAME - rename an object

❖ DML

Data Manipulation Language (DML) statements are used for managing data within schema objects. Some examples:

- ✓ SELECT - retrieve data from the a database
- ✓ INSERT - insert data into a table
- ✓ UPDATE - updates existing data within a table
- ✓ DELETE - deletes all records from a table, the space for the records remain
- ✓ MERGE - UPSERT operation (insert or update)
- ✓ CALL - call a PL/SQL or Java subprogram
- ✓ EXPLAIN PLAN - explain access path to data
- ✓ LOCK TABLE - control concurrency

❖ DCL

Data Control Language (DCL) statements. Some examples:

- ✓ GRANT - gives user's access privileges to database
- ✓ REVOKE - withdraw access privileges given with the GRANT command