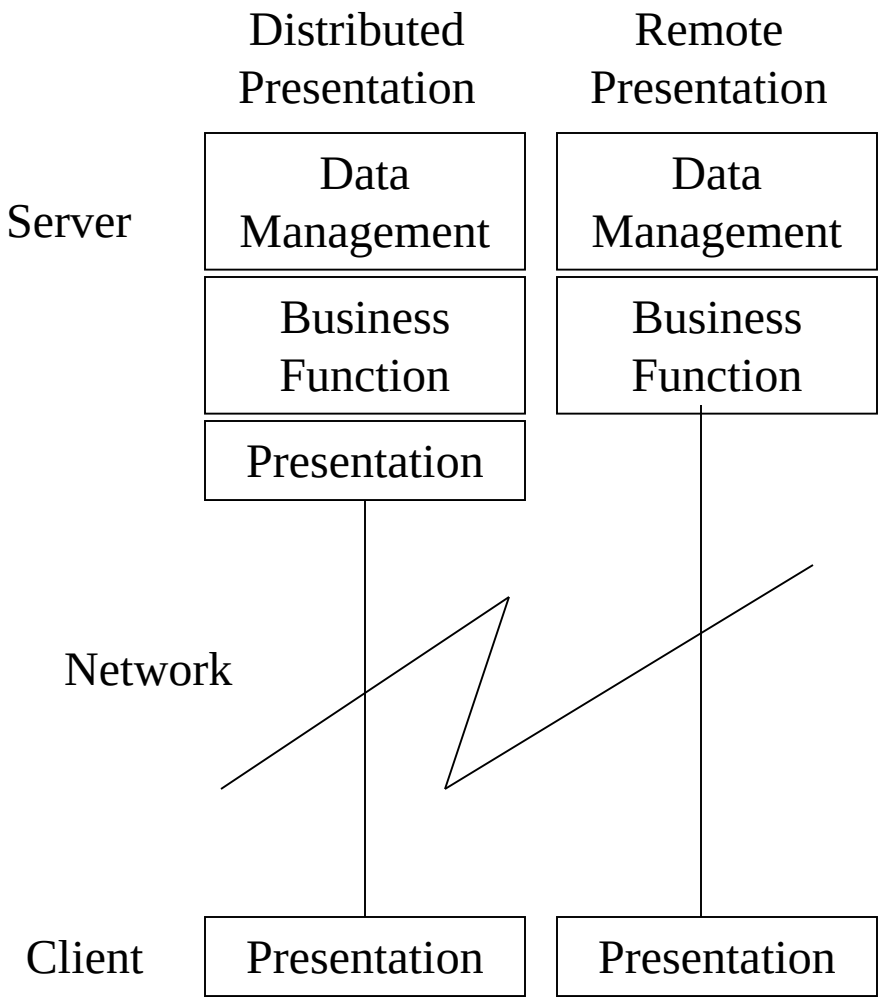
Client and Server

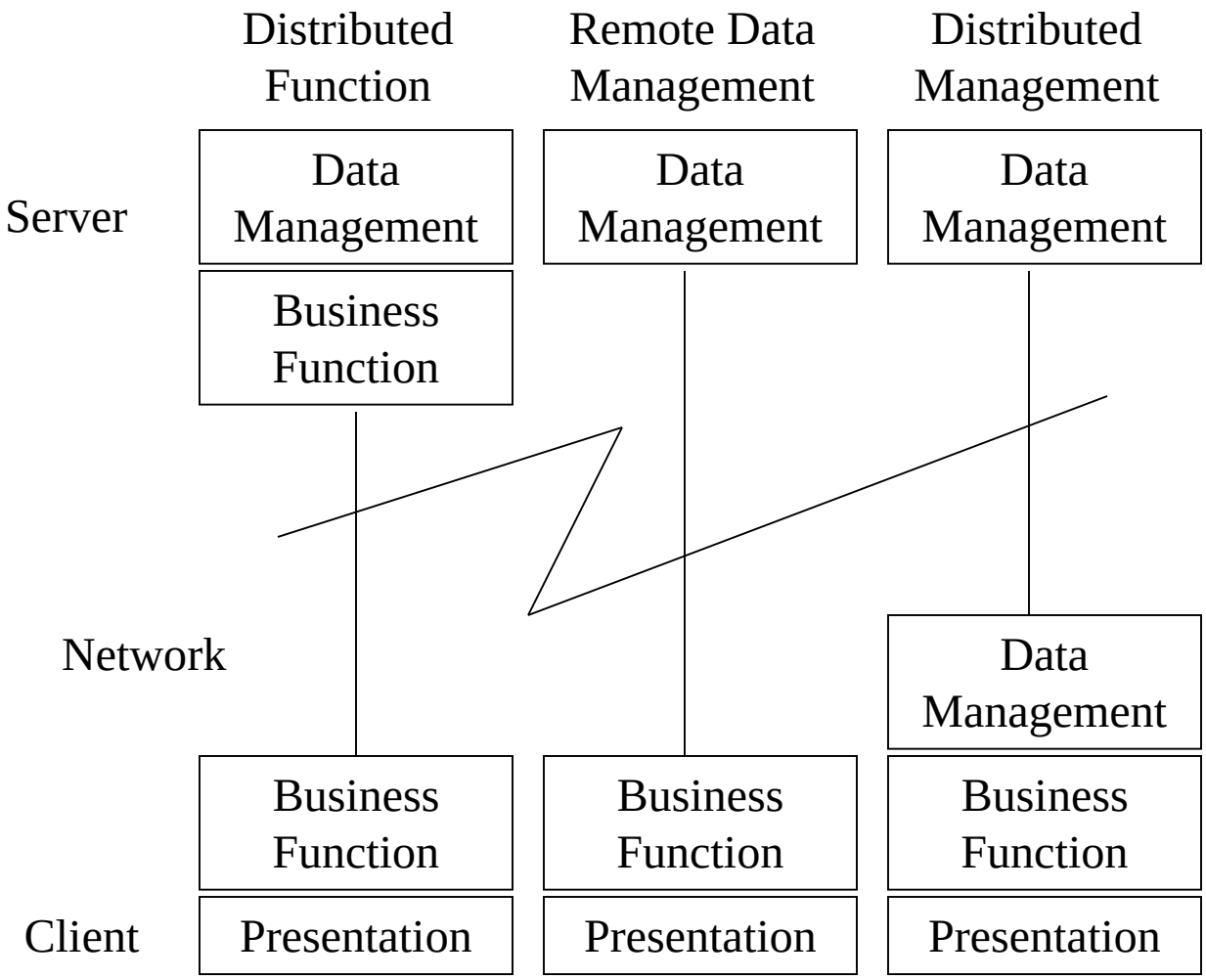
主從架構的沿革



傳統使用終端機模式：

- 終端機：沒有計算處理能力。
- 主機：負責執行、存取資料、印表等等處理過程。
- 集中式處裡的作業環境。
- 缺點：主機負擔重，處理速度快慢影響所有的共享使用者。

使用具有處理能力的 PC 模式



- 伺服器：處理檔案共享。
- 前端 PC：將想要執行的應用程式從伺服器下載並執行。
- 缺點：由於資料存取都必須完全由前端應用程式來控制，所以資料庫龐大時，資料的異動或查詢速度緩慢。

主從架構

- 觀念：分散式處理。
  - 分散「應用程式的處理程序」：把程式的處理功能切割成各小部分，然後再把這些不同工能的處理程序分散放置在不同的機器上，各自擔任不同的角色。
  - 分散「資料庫」：將龐大的資料庫依據資料性質的不同切割成幾個小型的資料庫，在分散放置網路上個個小型的伺服器上，提高資料庫處理的效能。

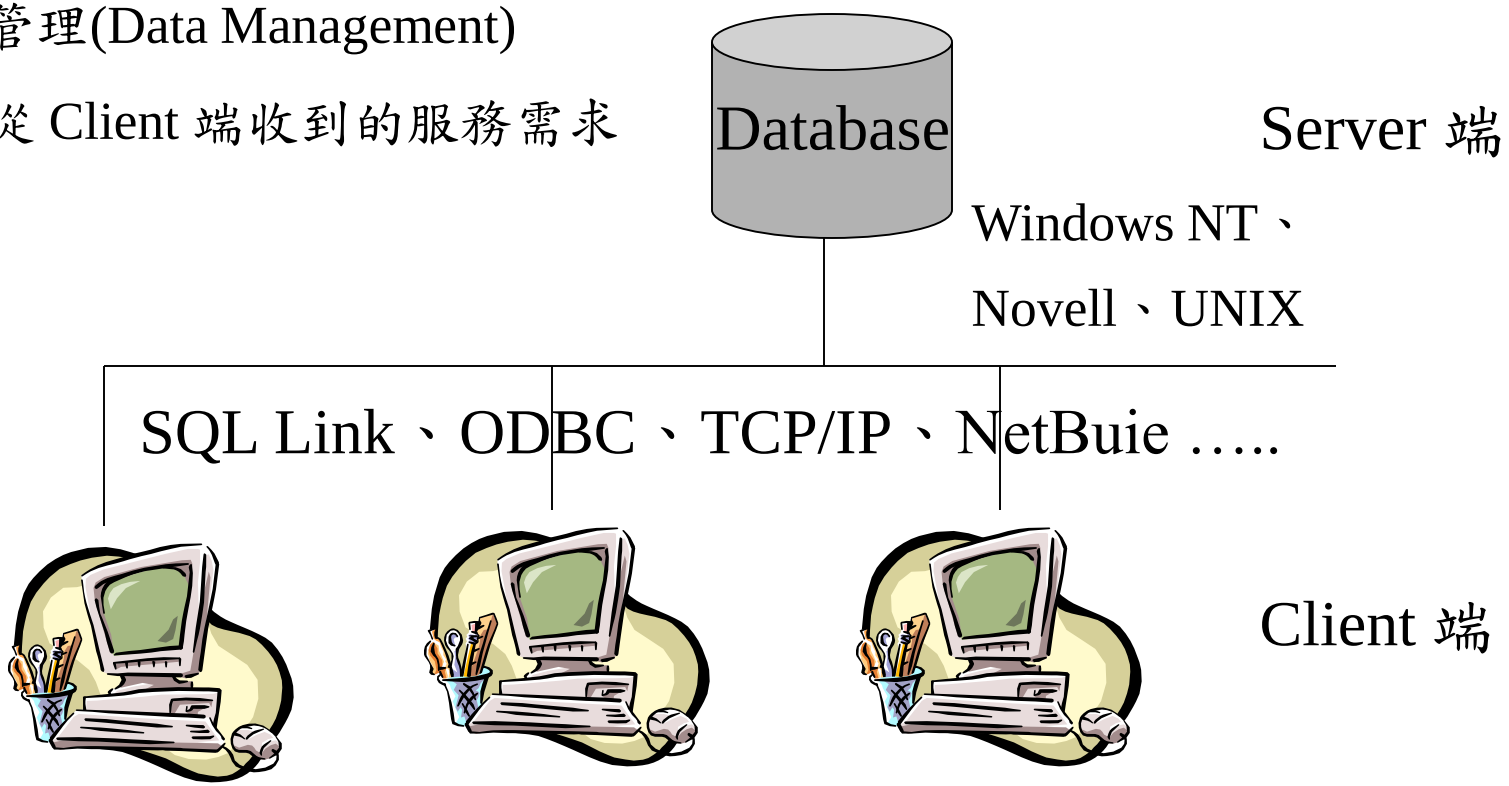
- 資料庫管理程式
  - 必須採用關聯式資料庫管理作業系統，如Oracle、Sybase、MS-SQL等，用來儲存，計算及處理資料。
- 前端應用程式
  - 必須能在圖形視窗介面的系統下執行。
- 精神：
  - 開放(open)、彈性(flexibility)。
  - 同一公司內包含有各種型式的電腦、網路、作業系統。

Server 端特性：

1.應用程式之邏輯處理(Business Function)

2.資料管理(Data Management)

3.回應從 Client 端收到的服務需求



Client 端特性：

1.處理使用者之介面(User Interface)

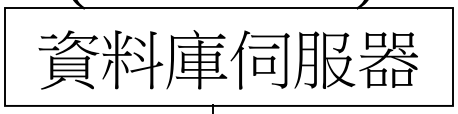
2.輸入資料之立即檢核(Data Valid)

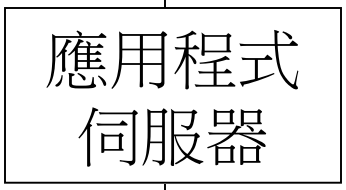
3.送出資訊需求給 Server 端(Query)

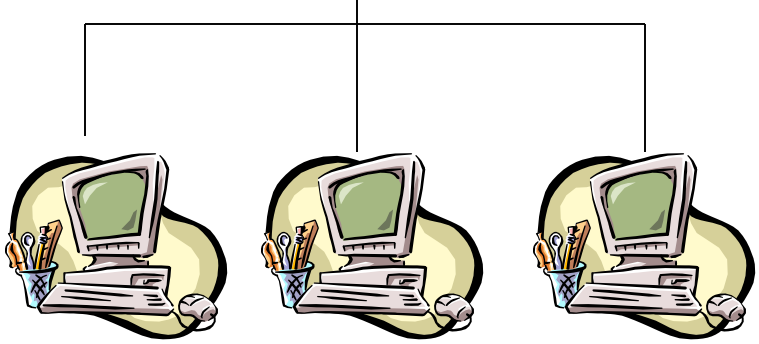
主從架構的組成元件

- Client
- Server
- 連接 Client 與 Server 的網路
- 建置分散式資料處理需考慮：
  - Client 端資料處理能力
  - Network 傳送資料負荷的能力
  - Server 端資料庫的功能與資料安全能力

三層式主從架構 (Multi-Tier)

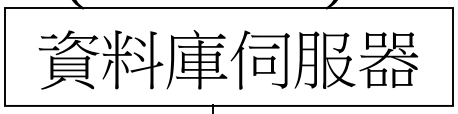


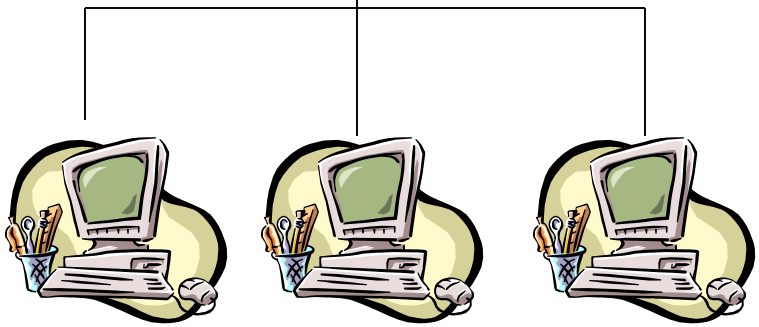




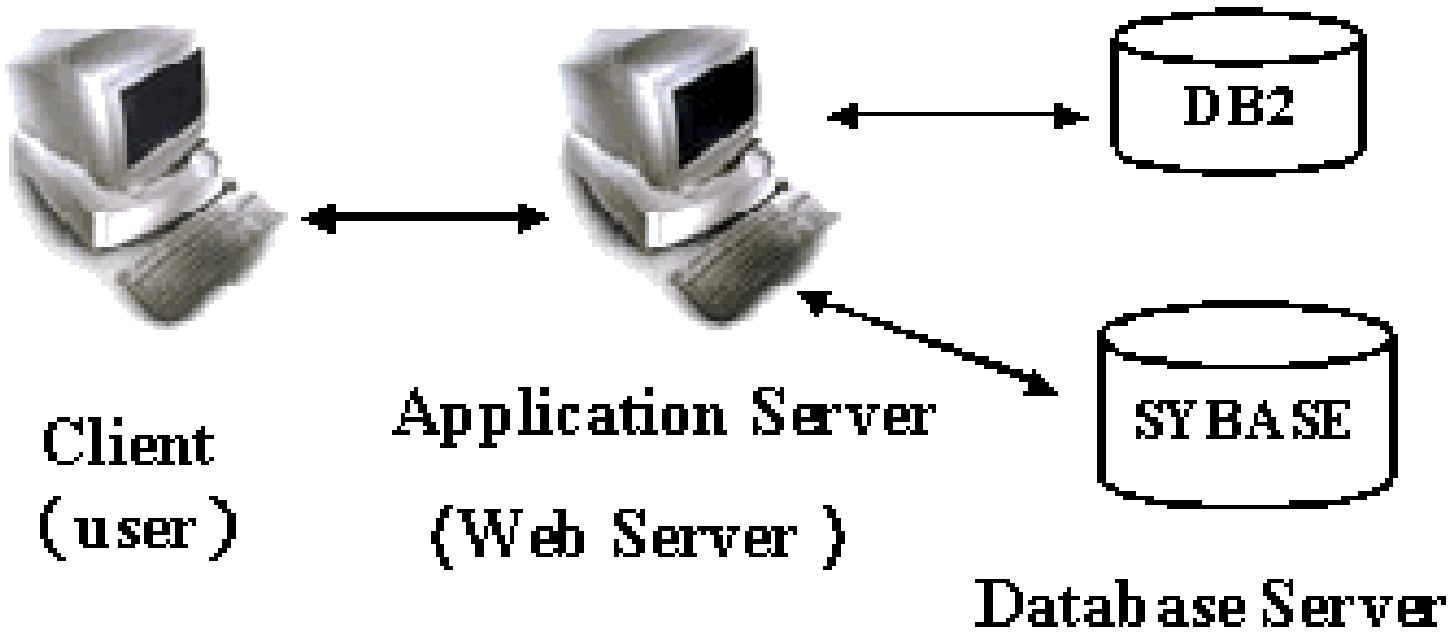
前端應用程式

二層式主從架構 (Two-Tier)



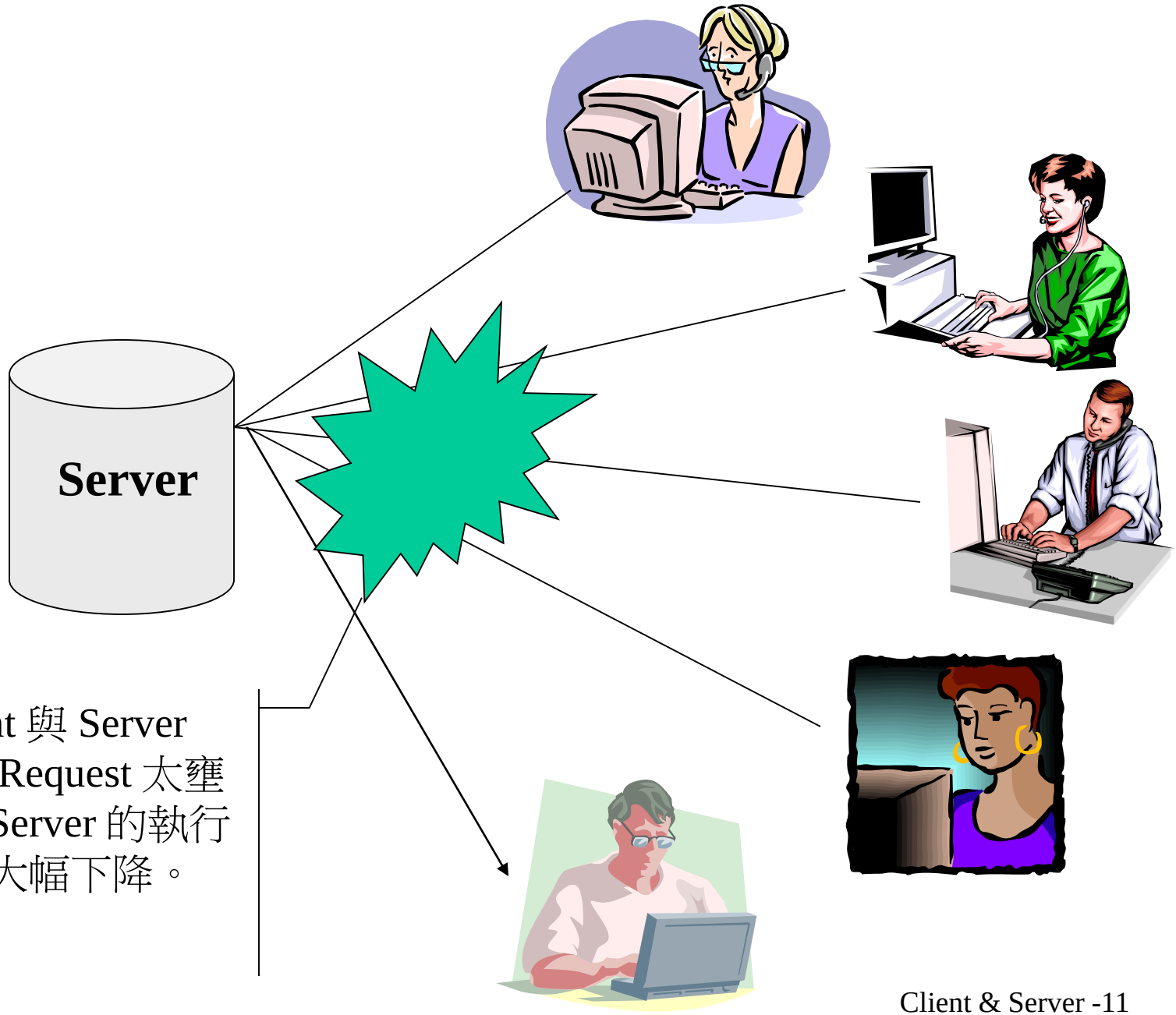


前端應用程式


DB2
SYBASE
Client
(user)
Application Server
(Web Server)
Database Server

Two-Tier

- 所謂 Two-Tier 是指整個系統架構只包含前端的應用程式與後端資料庫作業系統。
- 前端的應用程式：負責系統的圖形化的使用界面、使用者輸入資料的檢核、送出查詢的 SQL 命令給後端資料庫等工作。
- 後端的資料庫作業系統：負責接收並執行前端傳來的 SQL 命令，再依照需求存取存放實際資料的原始資料庫，最後將執行的結果回應給前端的應用程式。



當 Client 與 Server 之間的 Request 太壅塞時，Server 的執行效率將大幅下降。

Two-Tier

- 缺點：
  - 前端的使用者越多，前、後端之間的 Connection 就越多，後端伺服器的負擔就越重，進而大幅拖垮系統整體的運作效率。
  - 前端應用程式的分發工作麻煩，除了要把編譯產生的 .exe 執行檔複製到前端的電腦以外，還必須將一些 DLL 檔案複製到前端的電腦上，若前端公司有上百個分散各地，版本如何更新到這些前端的電腦？
  - 在 Two-Tier 的主從架構下，線上往往只有單一的資料庫伺服，若此伺服器當機，公司將運作停擺。

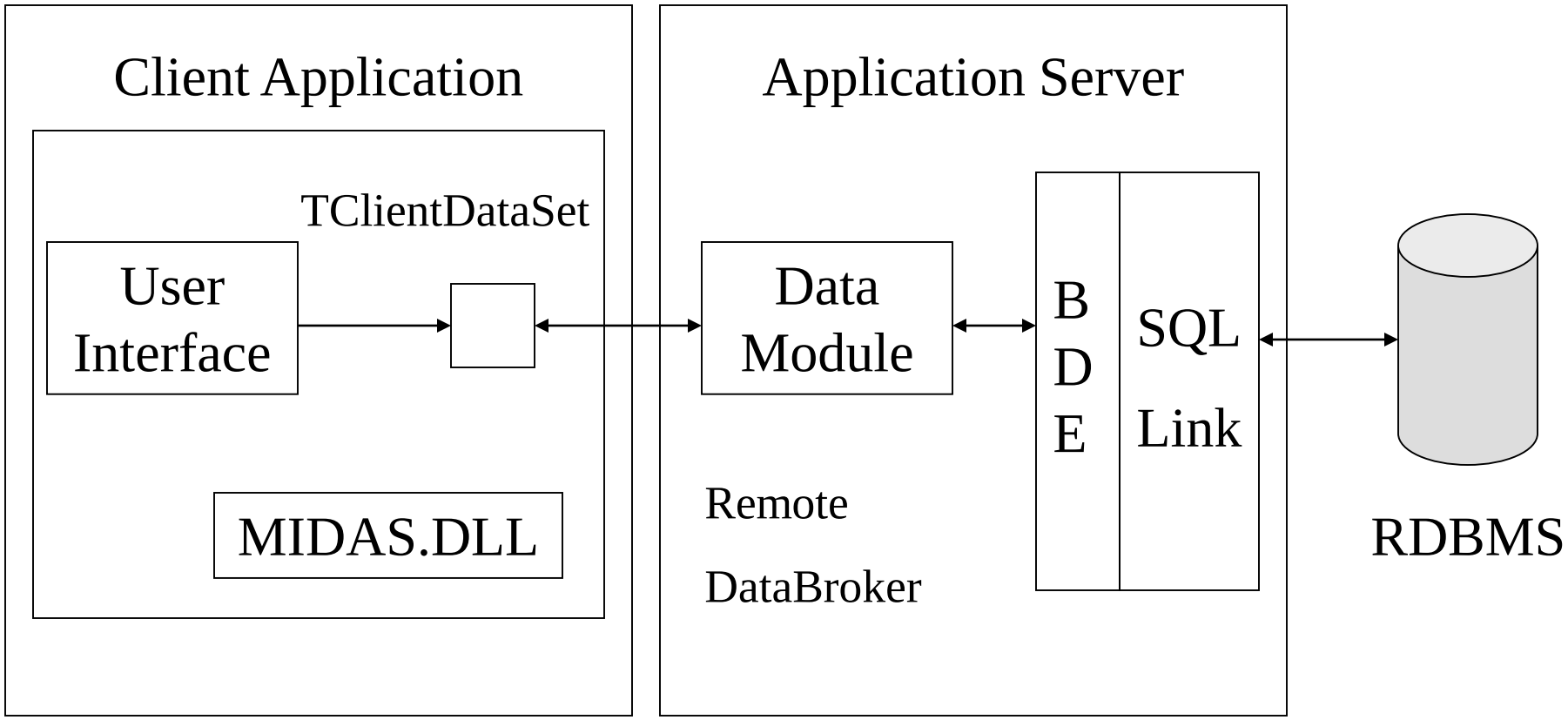
Three-Tier, Multi-Tier

- 架構：前端—應用程式伺服端—資料庫伺服端
- 應用程式伺服器：負責與前端的 Connection 處理，把前端電腦傳來的 Request 傳給資料庫伺服器，或是把資料庫伺服器傳回來的資料集(DataSet)回傳給前端的電腦。
- 資料庫伺服器：只要處理與應用程式伺服器之間的一個 Connection，其餘的時間可以專心地處理資料庫的存取動作，不再 Handle 與前端的 Connection，如此將大幅幾清資料庫伺服器的負擔，進而提升整工作效能。

Multi-Tier

- 解決 Two-Tier 的缺點：
  - Delphi 的 Multi-Tier 架構把原先 Two-Tier 安裝在前端的 DataModule、Link Drivers 都移到應用程式伺服器上，讓前端的應用程式只負責 User Interface 的處理動作，不需要再安裝其他軟體，只有一個單純的 EXE 檔（實際上，還必須包含相關的 DLL 檔案），所以當應用程式更新時，只要將修改後的執行檔透過網路複製到前端即可。若要修改 Link Drivers 的設定，只需在應用程式伺服器上改變即可。

- Multi-Tier 架構可以減少前端程式安裝困擾及日後維護容易



Multi-Tier

- 為了解決 Two-Tier 伺服器不穩導致系統停止運作的問題，Multi-Tier 架構提供一個 Business Broker 來管理應用程式伺服器的運作。
- Business Broker 的兩個基本功能：
  - 負載平衡 (Load Balancing)
    - 分散 Request 的 Process，不會被集中在某一台應用程式伺服器上。
  - 失敗回復 (Fail Over)
    - Business Broker 會判斷哪一台應用程式已經當機了，如果某台當機後，Business Broker 會把所有的 Request 丟給另外一台，不會產生系統停止運作的窘況。

Multi-Tier

- 在 Two-Tier 的應用程式中，經常會使用到一些商業邏輯原則，例如：保費率計算、薪資給付規則...等。在 Multi-Tier 的架構中，就把這些商業邏輯規則放在中間的應用伺服器上。