**I. Thông tin nhóm**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ tên** | **MSSV** | **Ghi chú** |
| 1 | Hà Thị Phương Thảo | 0912430 |  |
| 2 | Nguyễn Thị Thanh Thảo | 0912431 |  |
| 3 | Trương Nguyễn Thủy Tiên | 0912463 |  |
| 4 | Nguyễn Văn Tiến | 0912469 | Nhóm trưởng |

**II. Bảng phân công công việc**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Nội dung công viẹc** | **Thành viên 1**  **0912430** | **Thành viên 2**  **0912431** | **Thành viên 3**  **0912463** | **Thành viên 4**  **0912469** |
| 1 | Dịch tài liệu |  |  |  |  |
| 2 | Tổng hợp kết quả, viết tài liệu |  |  |  |  |

MỤC LỤC

[I. Động cơ thúc đẩy – Sự gia tăng của dữ liệu và tính phù hợp của thiết kế dữ liệu mức vật lý: 5](#_Toc308849223)

[II. Vòng đời dữ liệu: 6](#_Toc308849224)

[III. Các yếu tố của thiết kế vật lý: Index, Partition, và Cluster 8](#_Toc308849225)

*[1.](#_Toc308849226)**[Index:](#_Toc308849226)* [9](#_Toc308849226)

*[2.](#_Toc308849227)**[Khung nhìn cụ thể (Materialized view)](#_Toc308849227)* [10](#_Toc308849227)

*[3.](#_Toc308849228)**[Phân vùng và gom nhóm đa chiều (partitioning and multidimensional clustering)](#_Toc308849228)* [11](#_Toc308849228)

*[4.](#_Toc308849229)**[Những phương pháp khác đối với thiết kế dữ liệu vật lý](#_Toc308849229)* [11](#_Toc308849229)

[IV. Vì sao thiết kế dữ liệu mức vật lý lại khó khăn? 12](#_Toc308849230)

[V. Thủ thuật và nhận thức của các chuyên gia cơ sở dữ liệu 12](#_Toc308849231)

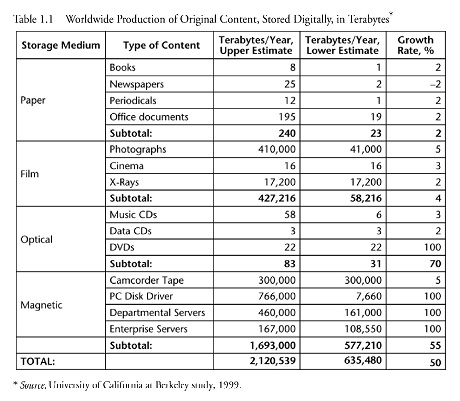
**GIỚI THIỆU VỀ THIẾT KẾ DỮ LIỆU MỨC VẬT LÝ**

*Thiết kế dữ liệu ở mức vật lý là thiết kế của những quan hệ và định nghĩa cơ bản đối với một hệ thống dữ liệu cụ thể, là một lĩnh vực lớn của các nhà thiết kế và lập trình ứng dụng.Thiết kế dữ liệu mức vật lý, sự sáng tạo của lưu trữ dữ liệu hiệu quả, và cơ chế phục hồi trên nền tảng đang sử dụng chính là lĩnh vực điển hình của một người quản trị cơ sở dữ liệu.*

# Động cơ thúc đẩy – Sự gia tăng của dữ liệu và tính phù hợp của thiết kế dữ liệu mức vật lý:

Thiết kế dữ liệu bị ảnh hưởng bởi khối lượng dữ liệu. Cơ sở dữ liệu với một vài dòng thật sự không có vấn đề gì với thiết kế dữ liệu vật lý, và hiệu suất của ứng dụng truy xuất vào một cơ sở dữ liệu nhỏ không thể bị ảnh hưởng nhiều bởi thiết kế dữ liệu ẩn bên dưới. Tuy nhiên, khi khối lượng dữ liệu càng tăng dần, cấu trúc của thiết kế dữ liệu vật lý bên dưới ngày càng trở nên quan trọng.

Một vài nhân tố đang thúc đẩy sự gia tăng nhanh chóng của dữ liệu ở cả 3 dạng: có cấu trúc (các bộ có quan hệ), bán cấu trúc (như XML), và không có cấu trúc (như audio, video). Phần nhiều sự tăng trưởng này là do sự mở rộng nhanh chóng và sử dụng rộng rãi của mạng lưới máy tính và trạm cuối (terminal) trong mỗi nhà, doanh nghiệp và cửa hàng trong thế giới công nghiệp.

Bảng 1.1 thể hiện dữ liệu từ một cuộc khảo sát năm 1999 được thực hiện bởi Đại học California ở Berkeley. Chúng ta có thể thấy trong cuộc khảo sát này, dữ liệu lưu trong đĩa từ gia tăng ở tỉ lệ 100% /năm đối với server doanh nghiệp và hành chính.

Còn một yếu tố đặc biệt nữa ảnh hưởng đến khối lượng dữ liệu, là giá cả của các thiết bị lưu trữ. Khoảng năm 2000, giá của thết bị lưu trữ giảm mạnh giúp người dùng có thể lưu trữ dữ liệu trên ổ đĩa máy tính hơn là trên giấy. Ngày nay số hóa các văn bản không chỉ là vấn đề quan tâm của việc lưu trữ và phân tích mà còn là vấn đề tài chính.

Mô hình tăng trưởng nhanh thay đổi khối lượng dữ liệu mà hệ thống cơ sở dữ liệu quan hệ phải truy xuất và thao tác, nhưng không thay đổi tốc độ một thao tác phải hoàn thành. Thực tế thì đối với mức độ lớn, mục đích thực thi đối với hệ thống xử lí dữ liệu được định dựa trên những gì con người mong muốn và khá độc lập với khối lượng dữ liệu được thực thi. Trong khi khối lượng dữ liệu và độ phức tạp trong phân tích đang tăng nhanh, thì những mong muốn đang thay đổi ở mức chậm hơn. Một vài sự giảm nhẹ được tìm thấy trong sức mạnh gia tăng của các server dữ liệu hiện đại, bởi vì khi khối lượng dữ liệu tăng, sức mạnh của máy tính phía sau cũng được tăng theo. Tuy nhiên, hiện tượng tăng sức mạnh xử lý được giảm nhẹ bằng nhu cầu củng cố công nghệ server để giảm chi phí IT, kết quả là khi server tăng sức mạnh xử lý, chúng ta thường dùng để tăng số lượng của mục đích hơn là dùng để thực hiện một mục đích nhanh hơn.

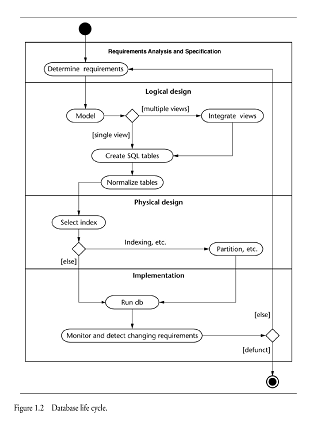
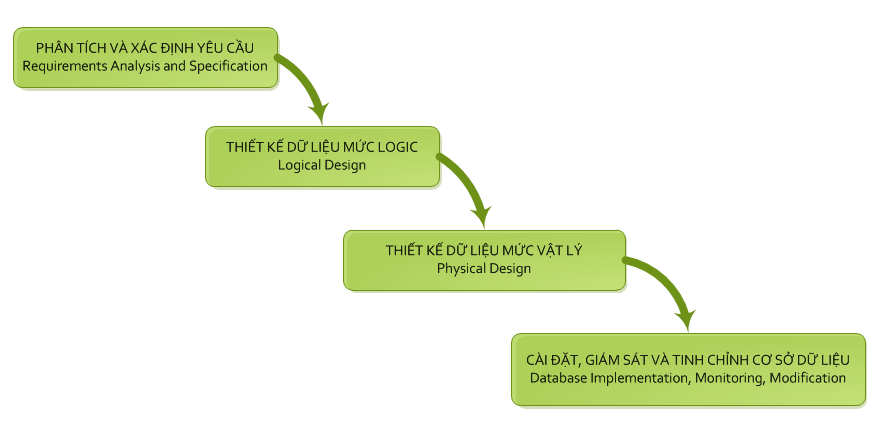
Cuối cùng, dữ liệu được sử dụng để tìm kiếm thông tin, không chỉ xử lí dữ liệu và sự nổi lên của xử lý phân tích online và khai thác dữ liệu và những dạng khác của máy tính mang trí tuệ kinh doanh sẽ dẫn đến sự gia tăng độ phức tạp trong những xử lý cần thiết.

Những yếu tố trên sẽ thúc đẩy nhu cầu tiếp cận thiết kế dữ liệu vật lý ở mức phức tạp và tinh vi.

# Vòng đời dữ liệu:

Vòng đời dữ liệu liên kết chặt chẽ với các bước trong thiết kế dữ liệu logic từ mô hình quan niệm của yêu cầu người dùng, thông qua hệ quản trị cơ sở dữ liệu (DBMS) xác định các bảng, cơ sở dữ liệu mức vật lý đã được sử dụng index, phân chia, cluster, và cụ thể hóa (materialized) có chọn lọc đến tối đa hiệu suất thật sự. Đối với một cơ sở dữ liệu phân tán, thiết kế vật lý còn liên quan đến cấp phát dữ liệu qua mạng máy tính. Khi thiết kế được hoàn thành, vòng đời còn tiếp tục với cài đặt và bảo trì cơ sở dữ liệu.

Vòng đời của dữ liệu được thể hiện trong hình 1.2. Thiết kế dữ liệu vật lý (bước 3) được định nghĩa trong ngữ cảnh toàn bộ vòng dời dữ liệu để thể hiện quan hệ của nó với các bước thiết kế khác



1. ***Phân tích yêu cầu:*** Yêu cầu dữ liệu được xác định bằng cách phỏng vấn người cung cấp và người sử dụng dữ liệu, và tạo ra một đặc tả yêu cầu hình thức. Đặc tả sẽ bao gồm dữ liệu cần cho xử lý, quan hệ tự nhiên của dữ liệu, và nền tảng phần mềm để cài đặt cơ sở dữ liệu.
2. ***Thiết kế dữ liệu mức logic:*** Thiết kế dữ liệu mức logic sẽ phát triển một mô hình quan niệm của dữ liệu từ tập các yêu cầu của người dùng và tinh chế mô hình đó thành các bảng SQL đã được chuẩn hóa. Mục tiêu của thiết kế logic là giữ lại được thực tế của thế giới người dùng trong phạm vi của các yếu tố dữ liệu và quan hệ của chúng, nhờ vào đó truy vấn và cập nhật dữ liệu có thể được lập trình dễ dàng. Lược đồ phổ quát, một khái niệm sơ đồ mô hình dữ liệu mức quan niệm thể hiện tất cả dữ liệu và quan hệ của chúng, được phát triển sử dụng các kĩ thuật như mô hình ER hoặc UML. Việc xây dựng mô hình dữ liệu phải được tích hợp vào một lược đồ phổ quát and chuyển thành các bảng SQL được chuẩn hóa. Bảng đã được chuẩn hóa là các bảng được phân rã hoặc phân chia thành những bảng nhỏ hơn để loại bỏ sự mất mát, ràng buộc dữ

***Lưu ý***: ở đây một vài nhà cung cấp công cụ cơ sở dữ liệu sử dụng thuật ngữ “mô hình logic” để ám chỉ mô hình dữ liệu mức quan niệm, và “mô hình vật lý” chỉ mô hình cài đặt chuyên biệt cho DBMS (ví dụ bảng SQL). Chúng ta cũng lưu ý nhiều mô hình dữ liệu mức quan niệm không chỉ đạt được từ những hỗn tạp, mà còn từ quá trình nghiên cứu đảo ngược từ một schema DBMS có sẵn.

1. ***Thiết kế dữ liệu mức vật lý:*** bước thiết kế dữ liệu mức vật lý liên quan đến việc chọn lựa index, phân chia, cluster, và chọn lựa cụ thể của dữ liệu. Thiết kế dữ liệu vật lý bắt đầu khi các bảng SQL được định nghĩa và chuẩn hóa. Nó tập trung vào các phương pháp lưu trữ và truy xuất các bảng đó trên đĩa, cho phép cơ sở dữ liệu thực thi với hiệu suất cao hơn. Mục tiêu của thiết kế vật lý là tối đa hiệu suất của cơ sở dữ liệu qua toàn bộ hình ảnh của các ứng dụng viết trên đó. Tài nguyên vật lý bao hàm cả thời gian trì hoàn trong thực thi ứng dụng dữ liệu bao gồm CPU, I/O (ví dụ đĩa), và mạng máy tính. Hiệu suất được đo bởi thời gian trì hoãn đáp ứng một câu truy vấn hoặc hoàn thành một cập nhật của ứng dụng độc lập, và còn bằng năng suất (throughput – in transactions per second) của toàn bộ hệ thống dữ liệu trên toàn bộ tập ứng dụng trong 1 đơn vị thời gian
2. ***Cài đặt, giám sát và tinh chỉnh cơ sở dữ liệu:*** Khi thiết kế logic và vật lý hoàn thành, dữ liệu được tạo ra thông qua việc cài đặt chính thức vào schema sử dụng ngôn ngữ DDL của DBMS. Sau đó ngôn ngữ DML có thể được sử dụng để truy xuất và cập nhật dữ liệu cũng như cài đặt index, thiết lập các ràng buộc như ràng buộc tham chiếu. Khi cơ sở dữ liệu bắt đầu thực thi, việc giám sát sẽ xác định yêu cầu hiệu suất có đạt được hay không. Nếu không đạt, điều chỉnh có thể được thực hiện để cải thiện hiệu suất. Những điều chỉnh khác có thể cần thiết khi yêu cầu thay đổi hoặc mong muốn của người dùng cuối gia tăng với hiệu suât tốt. Vì vậy, vòng đời tiếp tục với giám sát, thiết kế lại và điều chỉnh

# Các yếu tố của thiết kế vật lý:

Thiết kế dữ liệu vật lý bắt đầu với định nghĩa lược đồ của các record logic được tạo ra từ các pha thiết kế logic.

Trong lưu trữ, một record bao gồm con trỏ và record ở trên cần cho xác định và xử lí bởi hệ quản trị cơ sở dữ liệu. Một file là một tập hợp những record cùng loại được xây dựng tương tự nhau, và các bảng quan hệ được lưu trữ như file. Một cơ sở dữ liệu vật lý là tập hợp những record khác loại có quan hệ với nhau, có thể gồm tập hợp của các file có liên quan. Giao tác truy vấn và cập nhật được thực hiện hiệu quả bằng việc thực thi các phương thức tìm kiếm chính xác, chính là một phần của hệ quản trị cơ sở dữ liệu

## *Index:*

Một index là một tổ chức dữ liệu được thiết lập nhằm tăng tốc độ truy xuất dữ liệu từ bảng. Trong hệ quản trị cơ sở dữ liệu, index có thể được cài đặt bằng các lập trình viên cơ sở dữ liệu khi sử dụng các câu lệnh

**CREATE UNIQUE INDẼ supplierNum ON supplier (snum);**

Index có các dạng sau:

1. ***Unique Index****: Unique index là một cấu trúc dữ liệu (bảng) mà entry(record) của nó chứa các thuộc tính giá trị, các cặp con trỏ mà mỗi con trỏ sẽ chứa địa chỉ block của một record dữ liệu thật có liên quan đến giá trị thuộc tính như một giá trị index cốt lõi.* Điều này được biết như một index có thứ tự vì giá trị thuộc tính (khóa) trong index được sắp xếp theo giá trị mã ASCII. Nếu tất cả các giá trị khóa là kí tự thì thứ tự được đánh dấu theo alphabet. Index có thứ tự được lưu trữ trong cây B+ vì thế tìm kiếm dữ liệu trùng khớp nhanh. Khi giá trị khóa và block dữ liệu tương ứng được tìm thấy, sẽ có một bước truy xuất block chứa record mà bạn muốn, và một cuộc tìm kiếm nhanh trong bộ nhớ của block để tìm record
2. ***Index thứ cấp:*** Đôi khi truy vấn dữ liệu bằng thuộc tính sẽ tốt hơn là khóa, một thuộc tính điển hình sẽ có cùng giá trị trong nhiều record. Đối với thuộc tính không phải duy nhất, một index phải có nhiều thuộc tính, những cặp con trỏ cho các giá trị giống nhau, và mỗi con trỏ chứa 1 địa chỉ block của record có một trong các giá trị thuộc tính đó. Trong cây B+, nút lá sẽ chứa các thuộc tính, cặp con trỏ phải được tìm kiếm để dò tìm những record trùng khớp giá trị thuộc tính. Dòng lệnh SQL cho loại index này (còn được gọi là index thứ cấp) là:

**CREATE INDEX shippingDate ON shipment (shipdate);**

**/\*secondary index on non-key\*/**

Vì index thứ cấp thay đổi, có thể thiết lập một tập các giá trị thuộc tính mà bạn muốn sử dụng để truy vấn bảng. Mỗi entrry trong index sẽ bao gồm một tập các giá trị thuộc tính và một con trỏ tới block của record chứa chính xác những giá trị thuộc tính đó trong tập. Ví dụ:

**CREATE INDEX shipPart ON shipment (pnum, shipdate);**

**/\*secondary concatented index \*/**

Loại index này đặc biệt hiệu quả đối với truy vấn chứa cả một part number (pnum) và ngày shipping (shipdate). Đối với truy vấn liên quan chỉ một trong những thuộc tính trên, nó ít hiệu quả hơn vì kích thước của nó lớn hơn và do đó thời gian tìm kiếm lâu hơn.

1. ***Clustered index:*** Khi muốn cải thiện thời gian truy vấn một bảng dữ liệu, chẳng hạn như bảng truy vấn thông qua index nonunique trên shipdates, chúng ta có thể tổ chức dữ liệu sao cho các ship dates tương ứng có thể được lưu gần nhau (trên đĩa), và ship date nào gần nhau về nghĩa có thể được lưu gần nhau. Loại index này được gọi là ***clustered index***. Mặt khác còn có index được gọi là ***nonclustered index***. Chỉ có thể có 1 clustered index trên1 bảng bởi vì tổ chức vật lý của một bảng phải cố định
2. ***Hash index:*** Khi một bảng dữ liệu ở mức vật lý chưa được sắp xếp, nó có thể được tổ chức để hiệu quả truy xuất bằng cách sử dụng ***hash table index***, thường gọi là hash index. Loại index này căn cứ thường xuyên nhất trên khóa, nơi có giá trị duy nhất trong record dữ liệu. Giá trị thuộc tính (khóa) được chuyển qua một hàm ánh xạ nó với địa chỉ block bắt đầu, được gọi là bucket address. Bảng phải được cài đặt bằng cách insert tất cả record theo hàm hash, và sau đó sử dụng chính hàm hash đó để truy vấn dữ liệu sau này.
3. ***Bitmap index:*** thường được dùng cho index thứ cấp với nhiều giá trị thuộc tính, và được dùng trong cơ sở dữ liệu lớn trong data warehouse. Một bitmap index bao gồm một tập các bit vector mà mỗi bit tương ứng một giá trị thuộc tính cụ thể, và đối với mỗi record trong bảng, bit vector là “1” nếu record đó có giá trị bit vector, “0” nếu không có. Điều này sẽ có lợi nếu thuộc tính thưa thớt, nghĩa là thuộc tính có ít giá trị, như giới tính, loại khóa học. Nó sẽ không hoạt động tốt với các thuộc tính như họ tên, chức danh, tuổi ,… Bit vector có thể được lưu và truy xuất một cách hiệu quả, đặc biệt nếu chúng đủ nhỏ để được cấp phát trong bộ nhớ

## *Khung nhìn cụ thể (Materialized view)*

Khung nhìn trong SQL được lưu trữ trong định nghĩa hoặc template nhưng materialized view được lưu trong bảng giống như những bảng khác. Khung nhìn rất có ích để tăng tốc độ truy vấn dữ liệu được yêu cầu trước đó, hoặc các truy vấn dựa trên các tập hợp dữ liệu có thể được xây dựng trên materialized view để trả lời truy vấn thay vì phải quay lại dữ liệu ban đầu mỗi lần thực hiện. Tuy nhiên, cập nhật nhiều có thể làm cho materialized view ít hiệu quả hơn, và vấn đề này cần được chú trọng trong thiết kế và sử dụng

## *Phân vùng và gom nhóm đa chiều (partitioning and multidimensional clustering)*

***Phân vùng*** trong thiết kế dữ liệu vật lý là một phương pháp để giảm lượng công việc trên một thành phần phần cứng, như đĩa độc lập, bằng cách phân chia dữ liêu trên nhiều đĩa. Điều này có tác động trong việc cân bằng lượng công việc thông qua hệ thống và ngăn ngừa hiện tượng thắt cổ chai. Trong phạm vi phân vùng, giá trị thuộc tính được sắp xếp và phạm vi của giá trị được chọn lọc sao cho mỗi miền giá trị có số record bằng nhau. Những record trong miền xác định được cấp phát vùng đĩa riêng biệt vì thế chúng có thể được xử lý riêng biệt với các miền khác.

***Gom nhóm đa chiều*** (gom nhóm đa chiều – MDC) là một kỹ thuật mà dữ liệu có thể được gom nhóm theo các chiều như vị trí, thời gian, loai sản phẩm. Một cách chi tiết, MDC cho phép dữ liệu được gom nhóm bằng nhiều chiều cùng 1 lúc, ví vụ ván trượt tuyết được bán ở Winconsin suốt tháng 12. Tập hợp được dùng để tận dụng lượng công việc được biết và lấy trước trên dữ liệu

## *Những phương pháp khác đối với thiết kế dữ liệu vật lý*

***Nén dữ liệu*** là kỹ thuật cho phép dữ liệu vừa vặn trong một không gian cố định, vì thế truy xuất nhanh hơn nếu dữ liệu cần được quét nhiều lần. Chi phí nén được tính bằng thuật toán chuyển đổi dữ liệu ban đầu sang dạng nén để lưu trữ, và sau đó chuyển đổi từ dạng nén về dạng ban đầu.

***Phân chia dữ liệu (stripping)*** là kỹ thuật phân chia dữ liệu cần được truy xuất cùng nhau qua nhiều đĩa để đặt mức độ lớn hơn của xử lý song song và cân bằng tải, cả hai đều giúp lượng tải việc của hệ thống tăng lên và nói chung giảm thời gian truy vấn. Điều này đặc biệt phù hợp với kiến trúc đĩa theo mảng như RAID khi mà dữ liệu có thể được truy xuất song song trên nhiều đĩa theo cách tổ chức

Một cách khác cải thiện độ tin cậy của dữ liệu là kết hợp những kỹ thuật dư thừa dữ liệu như ***nhân bản (mirroring),*** trong đó dữ liệu sẽ được nhân đôi trên đĩa. Nhược điểm của phương pháp này là phải cập nhật bản copy của dữ liệu mỗi làn dữ liệu thay đổi, cũng như cần có không gian đĩa cần thiết.

***Tinh chế lược đồ:*** lược đồ tổng quát đôi lúc có thể được tinh chế lại bằng những cách nhất định để phản ánh yêu cầu xử lý nếu rõ ràng là sẽ đạt được nhiều hiệu quả nếu thực hiện. Điều này được gọi là denormalization. Nó bao gồm chọn ra những xử lí ưu thế (trội – có ảnh hưởng lớn) trên nền tảng của cường độ lớn, khối lượng lớn và độ ưu tiên ẩn, định nghĩa những mở rộng đơn giản của bảng có thể cải thiện hiệu suất truy xuất, đánh giá tổng chi phí cho truy vấn, cập nhật và lưu trữ, và xem xét tác động trên các mặt ví dụ như mất mát ràng buộc.

# Vì sao thiết kế dữ liệu mức vật lý lại khó khăn?

Thiết kế vật lý liên quan đến hàng trăm biến vốn rất khó để bắt kịp, đặc biệt là khi ảnh hưởng của nó thường tương quan với các giải pháp thiết kế khác nhau. Các tính toán độc lập về hiệu suất dựa trên các cơ chế index cho trước hoặc thuật toán phân vùng có thể ngốn hàng giờ thủ công và phân tích hiệu suất thường dựa trên so sánh giữa nhiều cấu hình và điều kiện công việc, vì vậy đòi hỏi hàng ngàn tính toán. Vì vậy mà thiết kế dữ liệu là vấn đề khó khăn.

# Thủ thuật và nhận thức của các chuyên gia cơ sở dữ liệu

***Thủ thuật 1:*** **The truth is out there, but you may not need it***. (Sự thật vẫn ở đó, nhưng bạn có thể không cần chúng)* Mỗi cơ sở dữ liệu đều có một thiết kế vật lý hoàn hảo, tối ưu về mặt lý thuyết. Nhưng trong thực tế không ai có thể tìm thấy vì tính phức tạp của tìm kiếm quá cao và những xử lý hợp lệ thì quá cồng kềnh. Thiết kế dữ liệu thật sự là một vấn đề khó. Tuy nhiên, sự phức tạp được giảm bớt bằng thực tế là cuối cùng điều gây khó khăn nhất không phải là hiệu suất của cơ sở dữ liệu có tốt được như trong lý thuyết hay không, mà là ứng dụng sử dụng cơ sở dữ liệu có thực hiện đủ tốt để đáp ứng nhu cầu người dùng hay không. Đủ tốt là một định nghĩa mập mờ. Trong hầu hết trường hợp, trong khi một thiết kế dữ liệu hoàn hảo thường thường khó nắm bắt, thì một thiết kế đạt thiết kế đạt hơn 85% tối ưu có thể đạt được bằng mere mortals (chí tử)

***Thủ thuật 2:*** **Be prepared to tinker*.*** *(Hãy sẵn sàng cho sự chấp vá)* Khả năng là vô tận, và bạn sẽ không bao giờ có thể khám phá hết. Nhưng với trí khôn và nhận thức về bạn và các vấn đề xung quanh khả năng bạn có thể tiến xa. Thử nghiệm và sai lầm là một phần của quá trình.

***Thủ thuật 3:*** **Use tools at your disposal***.(Hãy sử dụng công cụ có sắp đặt)* Xuyên suốt quyển sách này, chúng tôi sẽ miêu tả về các kỹ thuật và phương pháp đối với thiết kế dữ liệu mức vật lý. Nhiều thiết kế dữ liệu thực hiện thứ tự cường độ tệ hơn cả khả năng của nó, đơn giản vì người thiết kế không quan tâm đến việc sử dụng các kỹ thuật sẵn có. Thiết kế dữ liệu không phải bắt đầu và kết thúc với những chọn lựa index của cột đơn giản. Bằng cách khai thác những tính năng như điểu chỉnh bộ nhớ, materialized view, phân chia phạm vi, cluster đa chiều, cluster index hay phân vùng không chia sẻ (độc lập) bạn có thể cải thiện đáng kể một thiết kế dữ liệu đơn giản, đặc biệt là các xử lý truy vấn phức tạp