

# Tổng hợp Kiến thức Kỹ thuật Dữ liệu (Data Engineering)

Học viên Cao học KHMT Bách Khoa TP.HCM (HCMUT)

Ngày 31 tháng 12 năm 2025

## 1 Nguyên lý Phân tích & Thiết kế CSDL

### 1.1 Tổng quan Các giai đoạn Thiết kế

- Mức Quan niệm (Conceptual):** *Mục tiêu* — nắm bắt yêu cầu và ngữ nghĩa (độc lập với cài đặt); *Mô hình/Công cụ* — ER/EER; *Đầu ra* — lược đồ quan niệm (thực thể, thuộc tính, mối kết hợp, ràng buộc toàn vẹn).
- Mức Logic:** *Mục tiêu* — ánh xạ từ mức quan niệm sang mô hình DBMS đích (ví dụ: quan hệ); *Mô hình/Công cụ* — ánh xạ ER-sang-quan hệ, chuẩn hóa (FDs); *Đầu ra* — lược đồ quan hệ (bảng, khóa, ràng buộc toàn vẹn).
- Mức Vật lý:** *Mục tiêu* — xác định cấu trúc lưu trữ và đường dẫn truy xuất để tối ưu hiệu năng; *Mô hình/Công cụ* — phân tích tải, chỉ mục, tổ chức tập tin, băm; *Đầu ra* — lược đồ trong (cấu trúc lưu trữ, chỉ mục, đường dẫn truy xuất).

### 1.2 Nguyên lý Thiết kế Mức Quan niệm

- Phân tích Yêu cầu:** Làm việc với người dùng/chuyên gia nghiệp vụ để nắm bắt yêu cầu dữ liệu và yêu cầu chức năng (thao tác/giao dịch).
- Thành phần ER:** **Thực thể** (vd: NhanVien), **Thuộc tính** (đơn/phức hợp/đa trị/dẫn xuất), **Mối kết hợp** (sự liên kết giữa các thực thể).
- Ràng buộc Cấu trúc:** Tỷ số bản số (1:1, 1:N, M:N) và ràng buộc tham gia (*toàn phần* vs *từng phần*).
- Thực thể Yếu:** Được xác định thông qua **mối kết hợp xác định** với một thực thể **chủ** và một **khóa bộ phận**; thực thể yếu tham gia *toàn phần* vào mối kết hợp xác định.
- Tính chính Top-Down:** Tính chính lặp lại các thực thể tổng quát; áp dụng chuyên biệt hóa/tổng quát hóa (EER).

### 1.3 Nguyên lý Thiết kế Mức Logic

#### 1.3.1 Cơ bản về Mô hình Quan hệ

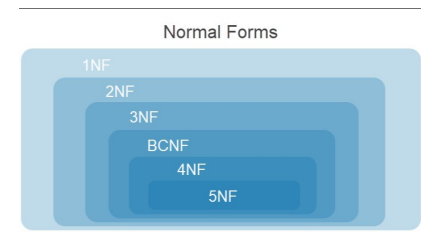
- Cấu trúc:** Lược đồ quan hệ  $R(A_1, \dots, A_n)$ ; các bộ (tuple) không có thứ tự và không cho phép trùng lặp trong mô hình hình thức.

- Ràng buộc Toàn vẹn:** Ràng buộc miền giá trị (nguyên tố, có kiểu), ràng buộc khóa (siêu khóa/khóa ứng viên/khóa chính), toàn vẹn thực thể (khóa chính không NULL), toàn vẹn tham chiếu (giá trị khóa ngoại phải xuất hiện trong khóa chính được tham chiếu).

#### 1.3.2 Lý thuyết Chuẩn hóa (Normalization Theory)

- Mục tiêu:** Giảm thiểu dư thừa, tránh dị thường (Thêm, Xóa, Sửa).
- Phụ thuộc Hàm (FD):**  $X \rightarrow Y$  (Nếu  $t_1[X] = t_2[X]$  thì  $t_1[Y] = t_2[Y]$ ).
- Các loại Key:**
  - Superkey:** Xác định duy nhất một bộ.
  - Candidate Key:** Superkey tối thiểu.
  - Prime Attribute:** Thuộc tính nằm trong bất kỳ Candidate Key nào.

#### Các Dạng Chuẩn (Normal Forms)



(Hierarchy:  $BCNF \subset 3NF \subset 2NF \subset 1NF$ )

- 1NF (Atomic):** Miền giá trị nguyên tố.  
→ *Vì phạm:* Thuộc tính đa trị, lồng nhau, lặp lại nhóm.
- 2NF (No Partial):** Là 1NF + Thuộc tính *non-prime* phụ thuộc đầy đủ vào khóa.  
→ *Vì phạm:*  $\exists X \subsetneq \text{Key}$  sao cho  $X \rightarrow \text{NonPrime}$ .  
(Chỉ xảy ra nếu Key là khóa phức hợp).
- 3NF (No Transitive):** Là 2NF + Không có phụ thuộc bắc cầu giữa các *non-prime*.  
→ *Định nghĩa:* Với mọi  $X \rightarrow A$  (không tầm thường), phải thỏa: (a)  $X$  là Superkey **HOẶC** (b)  $A$  là Prime Attribute.
- BCNF (Strict):** Nghiêm ngặt hơn 3NF.  
→ *Định nghĩa:* Với mọi  $X \rightarrow A$ ,  $X$  **bắt buộc là Superkey**.  
(Khác biệt: BCNF không chấp nhận ngoại lệ " $A$  là Prime" như 3NF).

#### Tính chất Phân rã (Decomposition Properties)

- Kết nối bảo toàn thông tin (Lossless Join):** (*Bắt buộc*) Để phân rã  $R$  thành  $R_1, R_2$  không bị mất dữ liệu, điều kiện là:  $(R_1 \cap R_2) \rightarrow R_1$  **HOẶC**  $(R_1 \cap R_2) \rightarrow R_2$ .  
(*Giao của 2 bảng phải là khóa của ít nhất 1 bảng*).
- Bảo toàn phụ thuộc (Dependency Preservation):** Các FD ban đầu có thể được kiểm tra riêng lẻ trên từng  $R_i$  mà không cần join lại. (BCNF có thể không bảo toàn phụ thuộc).

#### 1.3.3 Phi chuẩn hóa (Denormalization)

- Mục tiêu:** Cải thiện hiệu suất đọc bằng cách đưa dư thừa vào lược đồ, ngược với chuẩn hóa.
- Động lực:** Tránh Join tốn kém; giảm độ phức tạp truy vấn; tăng locality dữ liệu.

#### Kỹ thuật Denormalization:

- Materialized Views:** Kết quả truy vấn được tính trước và lưu trữ; cập nhật khi dữ liệu thay đổi.
- Precomputed Aggregates:** Lưu giá trị tổng hợp (COUNT, SUM) trong bản ghi để tránh tính lại.  
VD: *Lưu số email chưa đọc trong bảng User thay vì đếm mỗi lần.*
- Document Databases:** Nhúng (embedding) dữ liệu liên quan trong một document thay vì tham chiếu.  
VD: *MongoDB nhúng thông tin Worker vào document Project.*
- Star Schema (DW):** Dimension tables được denormalize (VD: Brand, Category trong dim\_product).
- Microservices:** Sao chép (replicate) dữ liệu giữa các service để tách biệt và giảm phụ thuộc.

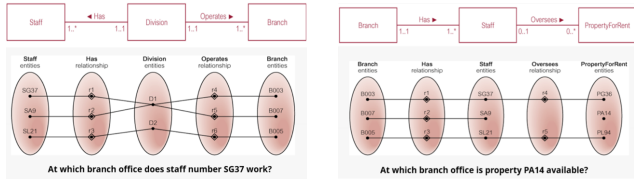
#### Đánh đổi (Trade-offs):

- Write Overhead:** Mỗi cập nhật phải sửa tất cả bản sao dư thừa → ghi chậm hơn, phức tạp hơn.
- Data Inconsistency:** Rủi ro không nhất quán nếu một bản sao được cập nhật mà bản khác thì không.
- Storage Cost:** Lưu trữ dữ liệu trùng lặp tốn bộ nhớ hơn.

*Khi nào dùng:* OLAP/DW (đọc nhiều), NoSQL (thiếu join), microservices (tách biệt). *Tránh:* OLTP cần ACID chặt.

#### 1.3.4 Bẫy Thiết kế CSDL (Design Traps)

- Bẫy Kết nối trong ER:**



(Fan Trap)

(Chasm Trap)

- **Fan Trap:** Đường dẫn giữa các thực thể mơ hồ do nhiều quan hệ 1:N phân nhánh từ một thực thể. VD: NhanVien → PhongBan → ChiNhanh (không xác định được nhân viên làm ở chi nhánh nào). *Giải pháp:* Thêm quan hệ trực tiếp NhanVien-ChiNhanh.
- **Chasm Trap:** Đường dẫn không tồn tại do tham gia tùy chọn. VD: KhanhHang → TaiSan → ChiNhanh (nếu tài sản chưa niêm yết thì không liên kết được khách hàng với chi nhánh). *Giải pháp:* Đổi sang tham gia bắt buộc hoặc thêm quan hệ trực tiếp.
- **Dị thường Cập nhật (Update Anomalies):** Do thiết kế không chuẩn hóa:
  - *Insertion:* Không thể thêm phòng ban mới nếu chưa có nhân viên.
  - *Deletion:* Xóa nhân viên cuối cùng làm mất thông tin phòng ban.
  - *Modification:* Thay đổi tên phòng ban phải cập nhật nhiều bộ.
- **Bộ giả (Spurious Tuples):** Kết nối các quan hệ phân rã sai (không qua PK/FK hợp lệ) tạo ra bản ghi ảo. *Giải pháp:* Dùng phân rã bảo toàn thông tin (lossless join).
- **Bẫy NULL:** Quá nhiều thuộc tính NULL lãng phí bộ nhớ và gây khó khăn trong truy vấn tổng hợp.
- **Sai lầm phổ biến:** Dùng PK của thực thể này làm thuộc tính của thực thể khác thay vì mô hình hóa quan hệ; gán PK vào thuộc tính của quan hệ; dùng thuộc tính đơn trị khi cần đa trị.
- **Entity Trap (Kiến trúc):** Thiết kế component 1-1 với bảng DB (VD: CustomerManager cho bảng Customer) thay vì theo workflow nghiệp vụ → vi phạm tách biệt dữ liệu trong microservices.

*Thực hành tốt:* Dùng BCNF/3NF; phân rã bảo toàn thông tin; đảm bảo đường dẫn ER rõ ràng; thiết kế theo hành vi nghiệp vụ, không theo thực thể.

## 1.4 Nguyên lý Thiết kế Mức Vật lý

- **Kiến trúc Lưu trữ:** Dữ liệu bền vững trên đĩa/SSD trong các khối (block) kích thước cố định.
- **Phân tích Tải (Job Mix):** Xác định các quan hệ/tập tin thường truy cập, điều kiện chọn (bằng/khác/khoảng), và tần suất cập nhật so với truy vấn.
- **Cấu trúc Chỉ mục:** Chỉ mục có thứ tự (B+-Trees) và chỉ mục băm; chỉ mục **chính/phân cụm** (quy định thứ tự vật lý; tối đa một trên mỗi tập tin) so với chỉ mục **phụ**.
- **Tối ưu hóa Truy vấn:** Dựa trên chi phí (thống kê) và các quy tắc kinh nghiệm (đẩy phép chọn/chiếu xuống sớm) để chọn kế hoạch hiệu quả.

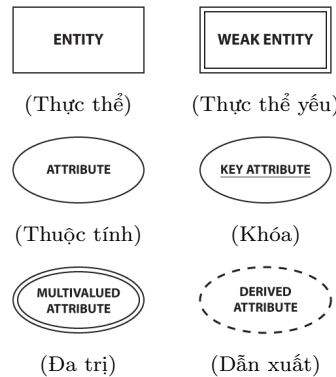
## 1.5 Các bước vẽ ERD (Steps to Create ERD)

1. **Xác định Thực thể (Entities):**  
Tìm các *danh từ* (Nouns) quan trọng trong yêu cầu (Vd: Employee, Student). Tránh nhầm lẫn thuộc tính là thực thể.
2. **Xác định Mỗi kết hợp (Relationships):**  
Tìm các *động từ* (Verbs) kết nối các thực thể (Vd: Works\_for, Teaches).
3. **Xác định Thuộc tính (Attributes):**  
Xác định thông tin chi tiết cho mỗi thực thể. Xác định thuộc tính đa trị, dẫn xuất, phức hợp.
4. **Xác định Khóa chính (Primary Keys):**  
Chọn thuộc tính định danh duy nhất cho mỗi thực thể và gạch chân nó.
5. **Xác định Bản số (Cardinality Ratio):**  
Phân tích số lượng tham gia: 1:1, 1:N, hay M:N.
6. **Xác định Ràng buộc tham gia (Participation):**  
Có bắt buộc không? (Total - Nét đôi) hay Tùy chọn? (Partial - Nét đơn).  
*Hỏi: "Thực thể A có thể tồn tại mà không cần B không?"*
7. **Vẽ phác thảo & Tinh chỉnh:**  
Vẽ sơ đồ, loại bỏ các thuộc tính dư thừa. Chuyển quan hệ M:N thành thực thể liên kết nếu cần thiết.

## 1.6 Ký hiệu Chen & EER (Chen Notation)

### 1.6.1 Thực thể & Thuộc tính (Entities & Attributes)

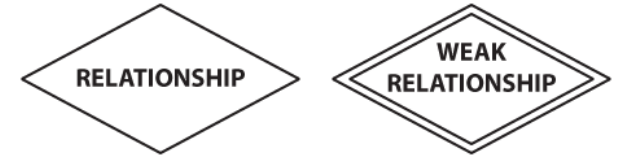
*Khi nào dùng:* **Thực thể** cho đối tượng độc lập (NhanVien, SanPham, KhanhHang); **Thực thể yếu** cho đối tượng phụ thuộc (NguoiPhuThuoc của nhân viên, ChiTietDonHang); **Khóa** là định danh duy nhất (MSNV, CCCD); **Đa trị** cho thuộc tính nhiều giá trị (số điện thoại, email); **Dẫn xuất** cho giá trị tính toán (tuổi từ ngày sinh, tổng tiền).



### 1.6.2 Mỗi kết hợp (Relationships)

*Khi nào dùng:* **Quan hệ thường** cho liên kết độc lập (NhanVien làm việc cho PhongBan, KhanhHang mua SanPham); **Quan hệ xác định**

khi thực thể yếu phụ thuộc vào thực thể chủ (NguoiPhuThuoc thuộc về NhanVien với khóa bộ phận là tên người phụ thuộc).

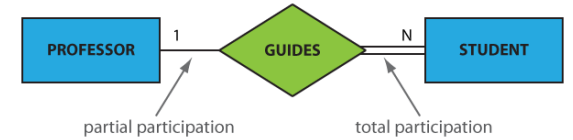


Bao gồm: Quan hệ (Hình thoi), Quan hệ xác định (Thoi đôi)

### 1.6.3 Ràng buộc (Constraints)

*Khi nào dùng:* Xác định quy tắc nghiệp vụ giữa các thực thể.

**Bản số (Cardinality):** 1:1 (NhanVien quản lý PhongBan - mỗi phòng có 1 trưởng), 1:N (PhongBan có NhanVien - nhiều nhân viên/phòng), M:N (NhanVien tham gia DUAN - nhiều-nhiều).  
**Tham gia (Participation):**

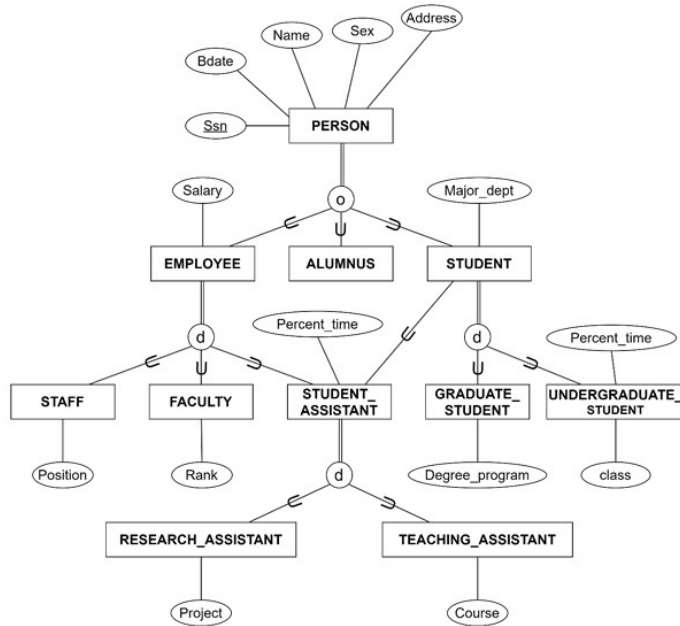


(Partial Participation (Từng phần): Nét đơn  
Total Participation (Toàn phần): Nét đôi)

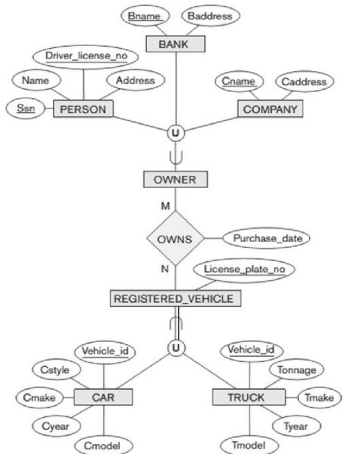
**Min-Max (min, max):** Ghi cặp số trên cạnh. Ví dụ: NhanVien (1, 1) làm việc cho (0, N) PhongBan nghĩa là mỗi nhân viên bắt buộc làm việc cho đúng 1 phòng ban, mỗi phòng ban có thể có 0 đến nhiều nhân viên. (0, 1): tùy chọn, tối đa 1; (1, N): bắt buộc, có thể nhiều. (An extension of Participation và Cardinality)

### 1.6.4 EER Chuyên biệt hóa & Tổng quát hóa

*Khi nào dùng:* **Disjoint** khi lớp con không chồng lấp (NhanVien là KỸ SƯ hoặc QUẢN LÝ, không đồng thời); **Overlapping** khi có thể thuộc nhiều lớp (NGƯỜI là SINH VIÊN và/hoặc NHÂN VIÊN); **Union** khi lớp con kế thừa từ nhiều lớp cha (CHỦ SỞ HỮU có thể là NGƯỜI hoặc CÔNG TY hoặc NGÂN HÀNG). **Total** khi mọi thực thể cha phải thuộc ít nhất 1 lớp con; **Partial** khi không bắt buộc.



(Disjoint & Overlapping)



(Union)

Ký hiệu: Hình tròn (d: disjoint, o: overlapping, U: union),  
Nét đôi (Total), Nét đơn (Partial).

## UNION Subclasses

A Union Subclass can be **total** or **partial**. A total category holds the **union** of all entities in its superclasses, whereas a partial category can hold a **subset of the union**. A total category is represented diagrammatically by a double line connecting the category and the circle, whereas a partial category is indicated by a single line.

## 2 Lưu trữ Dữ liệu & Chỉ mục

### 2.1 Cơ bản về Chỉ mục

#### 2.1.1 Chỉ mục là gì?

- **Mục đích:** Tăng tốc độ truy xuất dữ liệu bằng cách tạo đường dẫn phụ trợ đến các bản ghi.
- **Khóa tìm kiếm:** (Các) thuộc tính dùng để tìm bản ghi; không nhất thiết là khóa chính; có thể là khóa phức hợp (nhiều cột).
- **Đánh đổi:** Đọc nhanh hơn so với ghi chậm hơn (chi phí bảo trì chỉ mục khi INSERT/UPDATE/DELETE).

#### 2.1.2 Phân loại Chỉ mục

- **Theo Cấu trúc:**
  - *Có thứ tự (Ordered):* Các mục được sắp xếp (vd: B+-tree); hỗ trợ truy vấn khoảng.
  - *Băm (Hash):* Khóa được băm vào bucket; chỉ nhanh khi tìm kiếm chính xác (dấu bằng).
- **Theo Mật độ:**
  - *Đặc (Dense):* Một mục chỉ mục cho mỗi giá trị khóa tìm kiếm riêng biệt.
  - *Thưa (Sparse):* Một mục chỉ mục cho mỗi khối (hoặc mỗi giá trị phân cụm); nhỏ hơn, chi phí bảo trì thấp hơn.
- **Theo Thứ tự Vật lý:**
  - *Chỉ mục Chính/Phân cụm:* Khóa tìm kiếm quyết định thứ tự vật lý của tập tin (tối đa một per bảng); thường là chỉ mục thưa.
  - *Chỉ mục Phụ:* Đường dẫn truy cập thay thế độc lập với thứ tự vật lý; thường là chỉ mục đặc hoặc dùng gián tiếp cho các khóa không duy nhất.

## 2.2 B-Trees & B+-Trees

### 2.2.1 Tại sao dùng B+-Trees?

- **Mục tiêu:** Giảm thiểu I/O đĩa tốn kém bằng cách giữ chiều cao cây thấp thông qua hệ số rẽ nhánh (fan-out) cao.
- **Cấu trúc:** Cây cân bằng; nút trong chứa khóa + con trỏ con; nút lá chứa khóa + con trỏ dữ liệu (hoặc ID bản ghi) + liên kết đến lá kế tiếp.
- **Thao tác:** Tìm/thêm/xóa trong  $O(\log_{f_o} n)$  lần truy cập khối; tách/gộp nút để duy trì cân bằng.
- **B+- so với B-Tree:** B+- chỉ lưu con trỏ dữ liệu ở lá  $\Rightarrow$  hệ số rẽ nhánh nút trong cao hơn, quét tuần tự hiệu quả qua chuỗi liên kết lá.

### 2.2.2 Ví dụ Tính Dung lượng

Tham số: Kích thước khối  $B = 512$  bytes; kích thước khóa  $V = 9$  bytes; con trỏ dữ liệu  $Pr = 7$  bytes; con trỏ cây  $P = 6$  bytes.

### Bước 1: Tính Bậc (Số con trỏ tối đa mỗi nút)

- **Nút trong B-Tree:** Chứa  $p$  con trỏ cây +  $(p - 1)$  khóa +  $(p - 1)$  con trỏ dữ liệu.

$$(p \times 6) + ((p - 1) \times (7 + 9)) \leq 512$$

$$6p + 16p - 16 \leq 512 \Rightarrow 22p \leq 528 \Rightarrow p = 23$$

- **Nút trong B+-Tree:** Chứa  $p$  con trỏ cây +  $(p - 1)$  khóa (không có con trỏ dữ liệu).

$$(p \times 6) + ((p - 1) \times 9) \leq 512$$

$$6p + 9p - 9 \leq 512 \Rightarrow 15p \leq 521 \Rightarrow p = 34$$

- **Nút lá B+-Tree:** Chứa  $pleaf$  cặp khóa/con trỏ dữ liệu + 1 con trỏ kế tiếp.

$$(pleaf \times (7 + 9)) + 6 \leq 512$$

$$16 \times pleaf \leq 506 \Rightarrow pleaf = 31$$

### Bước 2: Ước lượng Tổng dung lượng (Đầy 69%)

- **B-Tree (3 mức):** Hệ số rẽ nhánh trung bình  $f_o = 23 \times 0.69 \approx 16$ .
  - Mức 0 (gốc): 15 mục, 16 con trỏ
  - Mức 1:  $16 \times 15 = 240$  mục
  - Mức 2:  $256 \times 15 = 3,840$  mục
  - **Tổng:**  $15 + 240 + 3,840 \approx 4,095$  mục
- **B+-Tree (3 mức):** Nút trong  $f_o = 34 \times 0.69 \approx 23$ ; Dung lượng lá  $= 31 \times 0.69 \approx 21$ .
  - Mức 0: 22 mục, 23 con trỏ
  - Mức 1:  $23 \times 22 = 506$  mục, 529 con trỏ
  - Mức lá:  $12,167 \times 21 \approx 255,507$  con trỏ dữ liệu
- **Nhận xét:** B+- chứa được  $\sim 4 \times$  số mục ở cùng chiều cao nhờ nút trong nhẹ hơn.

### 2.2.3 Các loại Chỉ mục Nâng cao

- **Chỉ mục Phức hợp:** Khóa nhiều cột (vd: (City, LastName)); hỗ trợ truy vấn tiền tố trái nhất (City), (City, LastName); sắp xếp cột theo độ chọn lọc.
- **Chỉ mục Dựa trên Hàm:** Chỉ mục trên biểu thức (vd: LOWER(email)); truy vấn phải dùng đúng hàm đó mới tận dụng được.

## 2.3 Chỉ mục Băm & Bitmap

### 2.3.1 Chỉ mục Băm (Hash Indexes)

- **Sử dụng:** Tìm kiếm chính xác cực nhanh (truy vấn điểm); không hỗ trợ khoảng.
- **Xử lý đụng độ:** Dùng danh sách liên kết (chaining) với bucket tràn.
- **Biến thể động:** Băm mở rộng/tuyến tính (Extendible/Linear hashing) tăng trưởng dần mà không cần xây lại toàn bộ.

### 2.3.2 Chỉ mục Bitmap

Tối ưu cho thuộc tính có độ chọn lọc thấp (ít giá trị riêng biệt).

- **Cấu trúc:** Đánh số thứ tự bản ghi (0, 1, 2, ...); mỗi giá trị riêng biệt có một bitmap; bit  $i = 1$  nếu bản ghi  $i$  có giá trị đó.

- **Ví dụ (bảng 5 dòng):**

- gender='m': 10010      gender='f': 01101

- income='L1': 11000      income='L2': 00100

- **Truy vấn:** gender='f' AND income='L2'  
01101 AND 00100 = 00100  $\Rightarrow$  bản ghi 2

- **Ưu điểm:** Gọn nhẹ (1 triệu dòng = 125 KB mỗi bitmap); thao tác bitwise nhanh; hiệu quả cho bộ lọc nhiều điều kiện; hỗ trợ COUNT qua đếm bit.

## 2.4 Tối ưu hóa Truy vấn & Phân tích Chi phí

### 2.4.1 Đo lường Chi phí & Thông tin Catalog

- **Độ đo chính (I/O):** Giảm thiểu chuyển khối (*b*) và truy cập ngẫu nhiên (seek *S*). Thời gian:  $b \times t_T + S \times t_S$ .
- **Kích thước quan hệ:**  $r$  = số bộ,  $b$  = số khối tập tin.
- **Chi tiết chỉ mục:**  $x$  = chiều cao chỉ mục đa mức (vd: B+-tree),  $b_{I1}$  = số khối chỉ mục mức 1.
- **Hệ số khối (Blocking factor):**  $bfr$  = số bộ trên mỗi khối.
- **Số lượng chọn (Selection cardinality):**  $s = sl \times r$  với  $sl$  là độ chọn lọc (selectivity).
- **Số giá trị phân biệt:**  $NDV(A)$  = số giá trị khác nhau của thuộc tính *A*.

### 2.4.2 Hàm Chi phí cho Phép Chọn

Phép chọn ( $\sigma$ ) có thể dùng quét tập tin hoặc truy cập chỉ mục/bấm tùy đường dẫn có sẵn. Chi phí chưa tính việc ghi kết quả.

#### S1: Tìm kiếm Tuyển tính (Vết cặn / A1)

- Trường hợp xấu nhất/không khóa: **C<sub>S1a</sub> = b**.
- Trung bình tìm bằng trên khóa: **C<sub>S1b</sub> = b/2** (dùng khi thấy).

#### S2: Tìm kiếm Nhị phân (tập tin có thứ tự)

**C<sub>S2</sub> = log<sub>2</sub> b + ⌈s/bfr⌉ - 1.**

#### S3a: Chỉ mục Chính (một bản ghi)

**C<sub>S3a</sub> = x + 1.**

#### S3b: Khóa Bấm (một bản ghi)

**C<sub>S3b</sub> = 1** (tính/tuyển tính) hoặc **2** (mở rộng).

#### S5: Chỉ mục Phân cụm (bằng trên không khóa / A3)

**C<sub>S5</sub> = x + ⌈s/bfr⌉.**

#### S6a: Chỉ mục Phụ (bằng trên không khóa / A4)

**C<sub>S6a</sub> = x + 1 + s** (trường hợp xấu nhất, bản ghi phân tán).

#### S6b: Chỉ mục Phụ (truy vấn khoảng)

**C<sub>S6b</sub> = x + (b<sub>I1</sub>/2) + (r/2).**

Lưu ý: Chi phí thời gian thường mô hình hóa là  $b \times t_T + S \times t_S$ , tách biệt truyền dữ liệu và tìm kiếm đầu từ.

### 2.4.3 Ví dụ: Phép chọn trên EMPLOYEE

Kích bản: EMPLOYEE có  $r_E = 10,000$ ,  $b_E = 2,000$ ,  $bfr_E = 5$ . Các chỉ mục/đường dẫn có sẵn:

- **Salary** (phân cụm, không khóa):  $x = 3$ ,  $s_{Salary} = 20$ .
- **Ssn** (phụ, khóa):  $x = 4$ ,  $s_{Ssn} = 1$ .
- **Dno** (phụ, không khóa):  $x = 2$ ,  $s_{Dno} = 80$  (từ 10,000/125).
- **Sex** (phụ, không khóa):  $x = 1$ ,  $s_{Sex} = 5,000$  (từ 10,000/2).

#### OP1: Tìm bằng trên Khóa

Truy vấn:  $\sigma_{Ssn='123456789'}(EMPLOYEE)$ .

- S1b (tuyển tính tb):  $C_{S1b} = b_E/2 = 1,000$ .
- S6a (chỉ mục phụ trên khóa):  $C_{S6a} = x_{Ssn} + 1 = 4 + 1 = 5$ .
- **Quyết định:** Chọn S6a ( $5 \ll 1,000$ ).

#### OP3: Tìm bằng trên Không khóa

Truy vấn:  $\sigma_{Dno=5}(EMPLOYEE)$ .

- S1a (tuyển tính):  $C_{S1a} = b_E = 2,000$ .
- S6a (phụ trên Dno):  $C_{S6a} = x_{Dno} + s_{Dno} = 2 + 80 = 82$ .
- **Quyết định:** Chọn S6a ( $82 \ll 2,000$ ). Nếu có chỉ mục phân cụm trên Dno:  $3 + \lceil 80/5 \rceil = 19$  khối.

#### OP4: Phép chọn Hội (Nhiều điều kiện)

Truy vấn:  $\sigma_{Dno=5 \wedge Salary > 30\,000 \wedge Sex='F'}(EMPLOYEE)$ . Bộ tối ưu so sánh các đường dẫn truy cập để lấy tập ứng viên ban đầu, sau đó kiểm tra các vị từ còn lại trong RAM.

- Qua Dno (S6a):  $C = x_{Dno} + s_{Dno} = 82$ .
- Qua khoảng Salary (phân cụm):  $C \approx x_{Salary} + (b_E/2) = 3 + 1,000 = 1,003$ .
- Qua Sex (S6a):  $C = x_{Sex} + s_{Sex} = 1 + 5,000 = 5,001$ .
- Vết cặn (S1a):  $C = 2,000$ .
- **Quyết định:** Dùng chỉ mục Dno (82), lấy 80 bộ, sau đó lọc  $Salary > 30,000$  và  $Sex='F'$  trong RAM.

### 2.4.4 Thuật toán Kết nối (Join) & So sánh Chi phí

#### Tham số Chính

- **Độ chọn lọc kết nối:**  $js = |R \bowtie S|/(|R||S|)$ ; với equi-join  $js \approx 1/\max(NDV(A), NDV(B))$ .
- **Số bộ kết quả:**  $jc = js|R||S|$ ; **chi phí ghi:**  $jc/bfr_{result}$  khối.
- **Bộ đệm:**  $n_B$  = số trang đệm khả dụng (ảnh hưởng chi phí nested-loop).

#### Kịch bản Thống nhất (Join)

EMPLOYEE ( $|E| = 10,000$ ,  $b_E = 2,000$ )  $\bowtie_{Dno=Dnumber}$

DEPARTMENT ( $|D| = 125$ ,  $b_D = 13$ ).

Chỉ mục trên *E.Dno* (phụ:  $x = 2$ ,  $s = 80$ ).

*Dnumber* là khóa chính ( $x = 1$ ).

Giả sử  $js = 1/125$ ,  $jc = 10,000$ ,  $bfr_{result} = 4$  (chi phí ghi = 2,500 khối),  $n_B = 3$ .

#### J1: Block Nested-Loop

Chi phí:  $C_{J1} = b_R + \left\lceil \frac{b_R}{n_B - 2} \right\rceil b_S + \frac{jc}{bfr_{result}}$ . Dùng DEPARTMENT làm vòng ngoài:  $C_{J1} = 13 + \lceil 13/1 \rceil \times 2,000 + 2,500 = 28,513$ .

### J2: Indexed Nested-Loop

- **DEPARTMENT ngoài  $\rightarrow$  EMPLOYEE trong:** Mỗi lần tìm  $= x + s = 2 + 80 = 82$ . Tổng =  $13 + 125 \times 82 + 2,500 = 12,763$ .
- **EMPLOYEE ngoài  $\rightarrow$  DEPARTMENT trong:** Mỗi lần tìm  $= x + 1 = 1 + 1 = 2$ . Tổng =  $2,000 + 10,000 \times 2 + 2,500 = 24,500$ .

### J3: Sort-Merge (Trộn sắp xếp)

Nếu đã sắp xếp:

$$C_{J3} = b_E + b_D + \frac{jc}{bfr_{result}} = 2,000 + 13 + 2,500 = 4,513$$

. Nếu chưa, cộng thêm chi phí sắp xếp ngoài mỗi quan hệ.

### J4: Partition-Hash (Bấm phân hoạch)

Chi phí xấp xỉ:  $C_{J4} \approx 3(b_E + b_D) + \frac{jc}{bfr_{result}} = 3 \times (2,000 + 13) + 2,500 = 8,539$ .

### Quyết định Kế hoạch

J3 (nếu đã sắp xếp) < J4 < J2 (D ngoài) < J2 (E ngoài) < J1. Ưu tiên Hash Join khi chưa sắp xếp; ưu tiên Sort-Merge khi đã có thứ tự.

## 2.5 Quy trình Tối ưu hóa Truy vấn

### 2.5.1 Tổng quan Quy trình

- Phân tích & Kiểm tra (Parse):** Kiểm tra cú pháp, tuân thủ lược đồ.
- Dịch (Translate):** Chuyển SQL sang đại số quan hệ (cây truy vấn).
- Tối ưu hóa Kinh nghiệm (Heuristic):** Áp dụng các quy tắc biến đổi.
- Tối ưu hóa Dựa trên Chi phí:** Liệt kê các kế hoạch, ước tính chi phí, chọn chi phí thấp nhất.
- Thực thi:** Hiện thực hóa kết quả trung gian hoặc pipeline kết quả.

### 2.5.2 Quy tắc Heuristic

- **Đẩy phép chọn ( $\sigma$ ) xuống:** Lọc sớm để giảm kích thước trung gian.
- **Đẩy phép chiếu ( $\Pi$ ) xuống:** Giảm chiều rộng bộ dữ liệu sớm.
- **Thay thế  $\sigma +$  Tích đề-các ( $\times$ ) bằng Kết ( $\bowtie$ ):** Tránh tích đề-các tồn kém.
- **Sắp xếp lại thứ tự kết:** Dùng tính giao hoán/kết hợp để tìm kết quả trung gian có lực lượng thấp.
- **Cây nghiêng trái (Left-deep trees):** Con phải luôn là bảng cơ sở  $\Rightarrow$  cho phép tìm kiếm chỉ mục, giảm không gian tìm kiếm.

### 2.5.3 Quyết định Dựa trên Chi phí

- **Chọn đường dẫn truy cập:** So sánh quét toàn bộ, chỉ mục phân cụm, chỉ mục phụ, bitmap cho mỗi vị từ.
- **Thuật toán kết nối:** Nested-loop (index/block), hash join, sort-merge dựa trên số lượng bộ và bộ nhớ.
- **Thứ tự kết nối:** Ước lượng độ chọn lọc kết nối  $js \approx 1/\max(NDV(A), NDV(B))$ ; số lượng  $jc = js \times |R| \times |S|$ .

2.6 Kỹ thuật Chuyên biệt

2.6.1 Cấu trúc Tối ưu Ghi

- **LSM-Tree:** Bộ đệm trong RAM (memtable) + các mức đĩa đã sắp xếp; ghi tuần tự; nén định kỳ; Bloom filter để bỏ qua các mức.
- **Buffer Tree:** Biến thể B-tree với bộ đệm ghi tại mỗi nút; gom nhóm các thay đổi xuống cây; độ trễ đọc tốt hơn LSM.

2.6.2 Truy cập Không gian & Đa khóa

- **Đa khóa:** Chỉ mục phức hợp cho truy vấn tiền tố; chỉ mục bao phủ (covering index) tránh tra cứu bảng.
- **Không gian:** R-tree (hình chữ nhật bao), kd-tree, quadtree cho dữ liệu địa lý/khoảng; hỗ trợ truy vấn vùng + lân cận gần nhất.

3 Big Data & Data Engineering

3.1 Khái niệm Cốt lõi (Core Concepts)

- **5V Big Data:**
  - **Volume:** Dung lượng lớn (TB → PB).
  - **Velocity:** Tốc độ sinh/xử lý nhanh (Streaming).
  - **Variety:** Đa dạng (Cấu trúc, bán cấu trúc JSON, phi cấu trúc).
  - **Veracity:** Độ tin cậy/nhiều của dữ liệu.
  - **Value:** Giá trị chuyển hóa thành lợi ích kinh doanh.
- **CAP Theorem:** Chỉ đạt được 2/3: **C**onsistency, **A**vailability, **P**artition Tolerance.  
→ *NoSQL thường chọn AP (Sẵn sàng + Chịu lỗi) thay vì CP.*
- **BASE:** Basically **A**vailable, **S**oft state, **E**ventual consistency (Nhất quán cuối cùng - chấp nhận dữ liệu cũ tạm thời).

3.2 Hệ sinh thái Hadoop (Hadoop Ecosystem)

Khung làm việc mã nguồn mở cho lưu trữ và xử lý phân tán.

- 1. HDFS (Storage):** Hệ thống tệp phân tán.
  - *NameNode:* Quản lý metadata (vị trí block).
  - *DataNode:* Lưu trữ block dữ liệu thực tế.
  - *Cơ chế:* Chia file thành block (128MB), replicate (x3) để chịu lỗi.
- 2. YARN (Resource Management):** "Hệ điều hành" của cluster.
  - Phân phối tài nguyên (RAM, CPU) cho các ứng dụng.

- Cho phép chạy nhiều engine (Spark, MapReduce) trên cùng 1 cụm.
- 3. MapReduce (Processing):** Mô hình xử lý Batch.
- *Map:* Xử lý song song cục bộ.
  - *Reduce:* Tổng hợp kết quả.

3.3 Công nghệ NoSQL (Storage Tech)

- **Document (MongoDB):** Schema-on-read (JSON/BSON). Linh hoạt, tốt cho CMS, Mobile apps. Query: `find()`.
- **Key-Value (Redis):** In-memory, cực nhanh (< 1ms). Dùng cho Caching, Session, Queue. Persist dạng Snapshot/AOF.
- **Wide-Column (Cassandra/HBase):** Ghi cực nhanh (LSM Tree). Tốt cho Time-series, Log, IoT. Query theo RowKey.
- **Graph (Neo4j):** Tối ưu cho quan hệ phức tạp (Mạng xã hội, Gợi ý, Fraud detection). Query: Cypher.

3.4 Xử lý Luồng (Streaming Processing)

Xử lý dữ liệu liên tục, độ trễ thấp (Real-time).

3.4.1 Các chiến lược xử lý (Strategies)

- **Thời gian (Time Domain):**
  - *Event Time:* Thời gian sự kiện xảy ra (quan trọng nhất).
  - *Processing Time:* Thời gian hệ thống nhận được dữ liệu.
  - *Watermark:* Cơ chế xử lý độ trễ (data đến muộn) trong Event Time.
- **Cửa sổ (Windowing):**
  - *Tumbling:* Cố định, không chồng (vd: mỗi 5p).
  - *Hopping/Sliding:* Có chồng lấp (vd: 5p, trượt mỗi 1p).
  - *Session:* Dựa trên hoạt động người dùng (hết timeout thì đóng).
- **Đảm bảo (Guarantees):**
  - *At-most-once:* Gửi 1 lần, chấp nhận mất (vd: Log).
  - *At-least-once:* Không mất, chấp nhận trùng lặp.
  - *Exactly-once:* Chính xác 1 lần (Khó nhất, cần Flink/Kafka).

3.5 Pipelines & Orchestration

3.5.1 Apache Kafka vs Airflow

Apache Kafka	Apache Airflow
<b>Loại:</b> Event Streaming Platform (Message Broker).	<b>Loại:</b> Workflow Orchestration (Quản lý quy trình).
<b>Đặc trưng:</b> <ul style="list-style-type: none"><li>• Log bền vững (Durable log).</li><li>• Decoupling (Tách rời).</li><li>• Replayable, High throughput.</li></ul>	<b>Đặc trưng:</b> <ul style="list-style-type: none"><li>• Code-as-infra (Python DAGs).</li><li>• Quản lý dependency phức tạp.</li><li>• Backfill (chạy lại quá khứ).</li></ul>
<b>Vai trò:</b> "Xương sống" vận chuyển dữ liệu <i>Real-time</i> .	<b>Vai trò:</b> "Nhạc trưởng" điều phối Job ( <i>Batch/ETL</i> ).

3.5.2 ETL vs ELT

- **ETL (Extract-Transform-Load):** Transform *trước* khi vào kho. Schema-on-write. Dữ liệu sạch, bảo mật. (Truyền thống).
- **ELT (Extract-Load-Transform):** Load raw vào kho *trước*, transform sau. Schema-on-read. Tận dụng sức mạnh Cloud DW (BigQuery, Snowflake). (Hiện đại).

3.6 Kho Dữ liệu (Data Warehousing)

3.6.1 Mô hình hóa (Modeling)

- **Star Schema:** Fact ở giữa, Dimension xung quanh. Phi chuẩn hóa dimension. *Hiệu năng cao, dễ query.*
- **Snowflake Schema:** Chuẩn hóa dimension (tách nhỏ). *Tiết kiệm không gian, join phức tạp.*

3.6.2 SCD (Slowly Changing Dimensions)

Tại sao cần? Để đảm bảo báo cáo lịch sử chính xác. Nếu KH chuyển từ HCM ra HN năm 2024, doanh số năm 2020 vẫn phải tính cho HCM.

- **Type 1 (Overwrite):** Ghi đè giá trị cũ. *Mất lịch sử.*
- **Type 2 (Add Row):** Thêm dòng mới + cột thời gian (Start/End Date) + cờ (is\_current). *Chuẩn nhất cho DW.*
- **Type 3 (Add Column):** Thêm cột "Previous Value". *Chỉ lưu được 1 giá trị cũ.*

3.6.3 DW vs DL vs Lakehouse

- **Data Warehouse:** Dữ liệu có cấu trúc, cho BI/Reporting.
- **Data Lake:** Dữ liệu thô (Raw), đa dạng, giá rẻ, cho ML/DS.
- **Lakehouse:** Kết hợp (Lưu trữ rẻ của Lake + Quản lý/ACID của Warehouse).