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

1. **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).
2. **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).
3. **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.
- **Tinh chỉnh Top-Down:** Tinh chỉnh lặp lại các thực thể tổng quát; áp dụng chuyên biệt hóa/tổng quát hóa (EER).

1.3 Nguyên lý Thiết kế Mức Logic

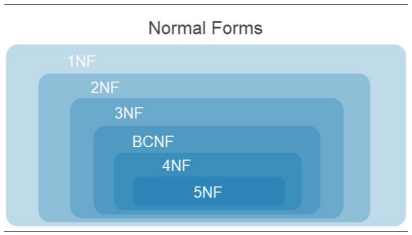
1.3.1 Cơ bản về Mô hình Quan hệ

- **Cấu trúc:** Lược đồ quan hệ $R(A_1, \ldots, A_n)$; 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):** $X \to Y$ (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 ⊂ 3NF ⊂ 2NF ⊂ 1NF)

1NF (Atomic): Miền giá trị nguyên tố.
↪ *Vi 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.
↪ *Vi phạm:* $\exists X \subsetneq Key$ sao cho $X \to NonPrime$.
(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 $X \to A$ (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 $X \to A$, X **bắt buộc là Superkey**.
(Khác biệt: BCNF không chấp nhận ngoại lệ "A là Prime"như 3NF).

Tính chất Phân rã (Decomposition Properties)

1. **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: $(R_1 \cap R_2) \to R_1$ **HOẶC** $(R_1 \cap R_2) \to R_2$.
   (Giao của 2 bảng phải là khóa của ít nhất 1 bảng).
2. **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.

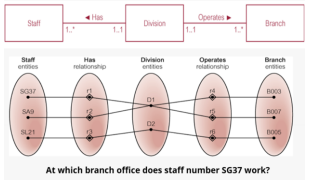
Đánh đổi (Trade-offs):

- **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.

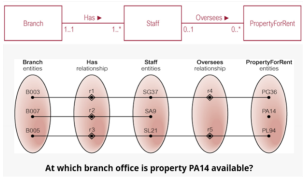
Khi nào dùng: OLAP/DW (đọc nhiều), NoSQL (thiếu join), microservices (tách biệt). *Tránh:* OLTP cần ACID chặt.

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.

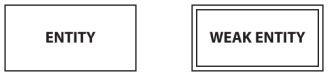
1.5 Các bước vẽ ERD (Steps to Create ERD)

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 Mối 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 (NguoiPhuThuoccủa 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).



(Thực thể)

(Thực thể yếu)

(Thuộc tính)

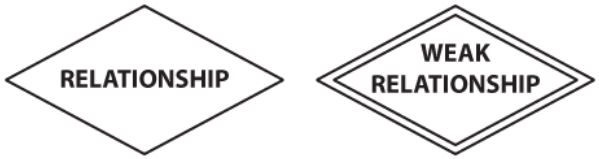
(Khóa)

(Đa trị)

(Dẫn xuất)

1.6.2 Mối 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).



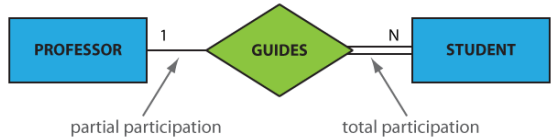
Bao gồm: Quan hệ (Hình thoi), Quan hệ xác định (Thoi đôi)

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):

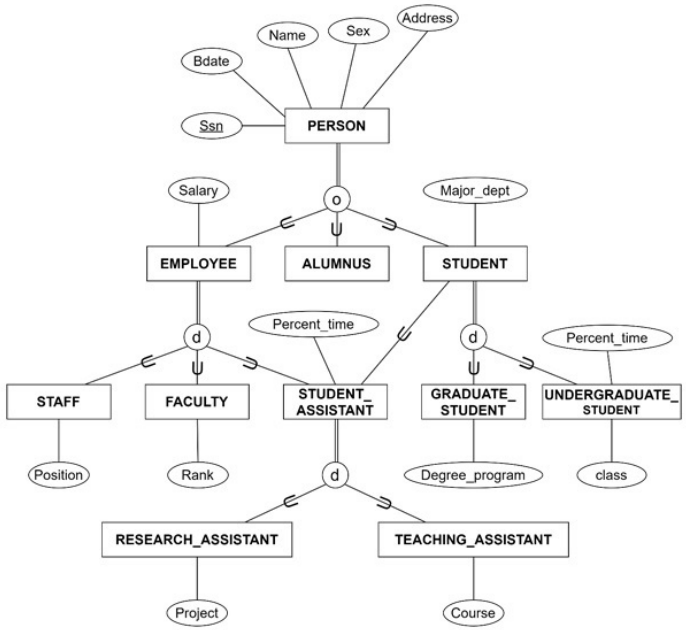


(Partial Participation (Từng phần): Nét đơn
Total Participation (Toàn phần): Nét đôi)

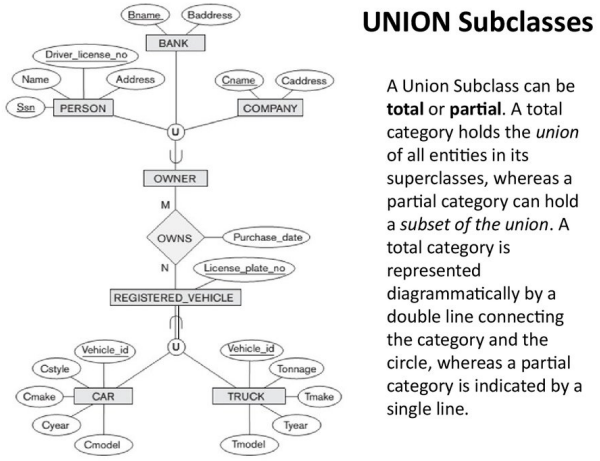
Min-Max (min, max): Ghi cặp số trên cạnh. Ví dụ: NhanVien $(1,1)$ *làm việc cho* $(0,N)$ PhongBan nghĩa là mỗi nhân viên bắt buộc làm việc cho đúng 1 phòng ban, mỗi phòng ban có thể có 0 đến nhiều nhân viên. $(0,1)$: tùy chọn, tối đa 1; $(1,N)$: bắt buộc, có thể nhiều. (An extension of Participation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

(Union)

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

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 $O(\log_{fo} n)$ 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 $\text{Pr} = 7$ 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.
  $(p \times 6) + ((p-1) \times (7+9)) \leq 512$
  $6p + 16p - 16 \leq 512 \quad \Rightarrow \quad 22p \leq 528 \quad \Rightarrow \quad p = 23$
- **Nút trong B+-Tree:** Chứa p con trỏ cây + $(p-1)$ khóa (không có con trỏ dữ liệu).
  $(p \times 6) + ((p-1) \times 9) \leq 512$
  $6p + 9p - 9 \leq 512 \quad \Rightarrow \quad 15p \leq 521 \quad \Rightarrow \quad p = 34$
- **Nút lá B+-Tree:** Chứa p_{leaf} cặp khóa/con trỏ dữ liệu + 1 con trỏ kế tiếp.
  $(p_{leaf} \times (7+9)) + 6 \leq 512$
  $16 \times p_{leaf} \leq 506 \quad \Rightarrow \quad p_{leaf} = 31$

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

- **B-Tree (3 mức):** Hệ số rẽ nhánh trung bình $fo = 23 \times 0.69 \approx 16$.
  - 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 $fo = 34 \times 0.69 \approx 23$; 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 (σ) có thể dùng quét tập tin hoặc truy cập chỉ mục/băm tùy đường dẫn có sẵn. Chi phí chưa tính việc ghi kết quả.

S1: Tìm kiếm Tuyến tính (Vét cạn / A1)

– Trường hợp xấu nhất/không khóa: $\mathbf{C_{S1a} = b}$.
– Trung bình tìm bằng trên khóa: $\mathbf{C_{S1b} = b/2}$ (dừng khi thấy).

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

$\mathbf{C_{S2} = \log_2 b + \lceil s/bfr \rceil - 1}$.

S3a: Chỉ mục Chính (một bản ghi)

$\mathbf{C_{S3a} = x + 1}$.

S3b: Khóa Băm (một bản ghi)

$\mathbf{C_{S3b} = 1}$ (tĩnh/tuyến tính) hoặc **2** (mở rộng).

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

$\mathbf{C_{S5} = x + \lceil s/bfr \rceil}$.

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

$\mathbf{C_{S6a} = x + 1 + s}$ (trường hợp xấu nhất, bản ghi phân tán).

S6b: Chỉ mục Phụ (truy vấn khoảng)

$\mathbf{C_{S6b} = x + (b_{I1}/2) + (r/2)}$.

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

Kịch 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: $\sigma_{\text{Ssn}='123456789'}(\text{EMPLOYEE})$.
– S1b (tuyến tính tb): $C_{S1b} = b_E/2 = 1{,}000$.
– S6a (chỉ mục phụ trên khóa): $C_{S6a} = x_{Ssn} + 1 = 4 + 1 = \mathbf{5}$.
– **Quyết định:** Chọn S6a ($5 \ll 1{,}000$).

OP3: Tìm bằng trên Không khóa

Truy vấn: $\sigma_{\text{Dno}=5}(\text{EMPLOYEE})$.
– S1a (tuyến tính): $C_{S1a} = b_E = \mathbf{2{,}000}$.
– S6a (phụ trên Dno): $C_{S6a} = x_{Dno} + s_{Dno} = 2 + 80 = \mathbf{82}$.
– **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: $\sigma_{\text{Dno}=5 \wedge \text{Salary}>30\,000 \wedge \text{Sex}='F'}(\text{EMPLOYEE})$. 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): $C = x_{Dno} + s_{Dno} = \mathbf{82}$.
– Qua khoảng Salary (phân cụm): $C \approx x_{Salary} + (b_E/2) = 3 + 1{,}000 = \mathbf{1{,}003}$.
– Qua Sex (S6a): $C = x_{Sex} + s_{Sex} = 1 + 5{,}000 = \mathbf{5{,}001}$.
– Vét cạn (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: $C_{J1} = 13 + \lceil 13/1 \rceil \times 2{,}000 + 2{,}500 = \mathbf{28{,}513}$.

J2: Indexed Nested-Loop

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

J3: Sort-Merge (Trộn sắp xếp)

Nếu đã sắp xếp:

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

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

J4: Partition-Hash (Băm phân hoạch)

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

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

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

2.5.2 Quy tắc Heuristic

– **Đẩy phép chọn (σ) xuống:** Lọc sớm để giảm kích thước trung gian.
– **Đẩy phép chiếu (Π) xuống:** Giảm chiều rộng bộ dữ liệu sớm.
– **Thay thế σ + Tích đề-các ($\times$) bằng Kết ($\bowtie$):** Tránh tích đề-các tốn 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 ⇒ 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 Đặc trưng Big Data (The Vs)

- **5V Cốt lõi (Core):**
  - **Volume:** Dung lượng khổng lồ (TB, PB, ZB).
  - **Velocity:** Tốc độ sinh ra & xử lý (Batch → Streaming).
  - **Variety:** Đa dạng định dạng (Structured, JSON, Video, Log).
  - **Veracity:** Độ tin cậy, tính xác thực (Messy/Noisy data).
  - **Value:** Giá trị chuyển hóa thành lợi ích kinh doanh.
- **Các V Mở rộng (Extended):**
  - **Variability:** Tính biến thiên (Ý nghĩa dữ liệu thay đổi theo ngữ cảnh/thời gian).
  - **Validity:** Tính hợp lệ (Dữ liệu có đúng định dạng/chuẩn để dùng không).
  - **Vulnerability:** Tính bảo mật (Dễ bị tấn công/rò rỉ).
  - **Volatility:** Độ bay hơi (Thời gian lưu trữ trước khi xóa/lưu trữ lâu dài).
  - **Visualization:** Khả năng trực quan hóa (Để con người hiểu được).
- **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: B**asically **A**vailable, **S**oft state, **E**ventual consistency (Nhất quán cuối cùng - chấp nhận dữ liệu cũ tạm thời).

3.2 Paradigm: Batch vs Streaming

Hai mô hình xử lý dữ liệu cơ bản trong Big Data.

	Batch Processing	Streaming Processing
Đặc trưng	Xử lý dữ liệu tĩnh, lượng lớn theo lô.	Xử lý dữ liệu động, liên tục theo thời gian thực.
Độ trễ	Cao (minutes - hours).	Thấp (seconds - milliseconds).
Công cụ	Hadoop MapReduce, Apache Spark (Batch mode).	Apache Flink, Spark Streaming, Kafka Streams.
Use Case	ETL, báo cáo cuối ngày, ML training.	Real-time analytics, fraud detection, monitoring.
Ưu điểm	Xử lý hiệu quả khối lượng lớn, đơn giản.	Phản hồi nhanh, phát hiện sự kiện ngay lập tức.
Nhược điểm	Không real-time, lãng phí khi data nhỏ.	Phức tạp, khó debug, cần xử lý out-of-order.

Kiến trúc Lai (Hybrid):

- **Lambda Architecture:** Batch layer (chính xác) + Speed layer (real-time) + Serving layer. Phức tạp, duy trì 2 code base.
- **Kappa Architecture:** Chỉ dùng Streaming (đơn giản hóa). Mọi dữ liệu qua stream processor, replay từ Kafka khi cần.

3.3 Phân mảnh & Sao chép (Partitioning & Replication)

3.3.1 Sao chép (Replication)

Mục đích: High Availability (HA) và giảm độ trễ đọc.

- **Single-Leader (Master-Slave):** Mọi ghi vào Leader, Leader chép sang Followers. *Dễ nhất quán, nhưng Leader là nút cổ chai.*
- **Multi-Leader:** Nhiều node chấp nhận ghi. *Tốt cho đa trung tâm dữ liệu, nhưng khó xử lý xung đột.*
- **Leaderless (Dynamo-style):** Ghi/Đọc gửi tới nhiều node. Dùng cơ chế **Quorum** để xác nhận:
  $w + r > n$ (Write nodes + Read nodes > Total replicas) → Đảm bảo đọc thấy dữ liệu mới nhất.

3.3.2 Phân mảnh (Partitioning/Sharding)

Mục đích: Scalability (Mở rộng dung lượng/băng thông).

- **Key Range Partitioning:** Chia theo khoảng khóa (A-C, D-F).
  → *Ưu:* Query theo khoảng (Range scan) hiệu quả.
  → *Nhược:* Dễ bị *Hotspot* (nếu user dồn vào vần A).
- **Hash Partitioning:** Băm khóa để chia đều ($hash(key)\%N$).
  → *Ưu:* Phân phối đều, tránh Hotspot.
  → *Nhược:* Mất khả năng Range Query (phải quét tất cả).

3.4 Định dạng Lưu trữ (File Formats)

Lựa chọn định dạng ảnh hưởng trực tiếp đến hiệu năng đọc/ghi.

3.4.1 Row-based vs. Column-based

- **Row-oriented (CSV, Avro):**
  - Lưu trữ tuần tự từng dòng.
  - *Ưu điểm:* Ghi nhanh (append), tốt khi truy xuất toàn bộ thông tin của 1 entity (OLTP).
  - *Nhược điểm:* Chậm khi tính toán tổng hợp (SUM, AVG) vì phải đọc cả dữ liệu không cần thiết.
- **Column-oriented (Parquet, ORC):**
  - Lưu trữ riêng biệt từng cột.
  - *Ưu điểm:* Nén cực tốt (do dữ liệu cùng kiểu), tối ưu cho OLAP (chỉ đọc cột cần thiết).
  - *Nhược điểm:* Ghi chậm, update tốn kém.

3.4.2 So sánh Avro, Parquet, ORC

Đặc điểm	Avro	Parquet
Mô hình	Row-based	Column-based
Schema	JSON (lưu trong file)	Binary (footer)
Tối ưu cho	Ghi nhiều (Write heavy)	Đọc nhiều (Read heavy)
Schema Evo	Rất tốt (Thêm/bớt field)	Hạn chế
Ecosystem	Kafka, Hadoop	Spark, Impala, Presto

3.5 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 - chia để trị trên cụm phân tán.
   - *Map:* Chia nhỏ & Gán nhãn. Input → Split → `<Key, Value>`.
   - *Shuffle:* Xáo trộn & Gom nhóm. Chuyển dữ liệu qua mạng, gom cùng Key.
   - *Reduce:* Tổng hợp. Xử lý danh sách Value của mỗi Key.

3.6 Công nghệ NoSQL (Storage Tech)

Các mô hình NoSQL cho use case khác nhau.

MongoDB (Document Store)

- **Mô hình:** Schema-on-read, lưu trữ JSON/BSON documents. Collections thay vì tables.
- **Ưu điểm:** Linh hoạt schema (mỗi doc có cấu trúc khác nhau), dễ scale horizontal (sharding), query mạnh (aggregation pipeline).
- **Architecture:** Replica Sets (HA), Sharding (scale-out), WiredTiger storage engine.

- **Use Case:** CMS, Mobile apps, Catalog, Real-time analytics. *VD: Forbes, eBay, Uber.*
- **Trade-offs:** Không ACID cross-document (trước v4.0), chiếm RAM nhiều.

Redis (Key-Value Store)

- **Mô hình:** In-memory key-value, data structures (String, Hash, List, Set, Sorted Set).
- **Ưu điểm:** Cực nhanh (< 1ms latency), atomic operations, support pub/sub, Lua scripting.
- **Persistence:** RDB (snapshot) hoặc AOF (append-only log). Có thể dùng cả 2.
- **Use Case:** Caching (session, query result), Message Queue (Celery), Leaderboard, Rate limiting. *VD: Twitter, GitHub, Stack Overflow.*
- **Trade-offs:** Giới hạn RAM, single-threaded (1 core), không có query phức tạp.

Cassandra (Wide-Column Store)

- **Mô hình:** Wide-column, mỗi row có thể có số cột khác nhau. Organize theo Column Family.
- **Ưu điểm:** Ghi cực nhanh (LSM Tree), linear scalability, masterless (P2P), multi-datacenter replication.
- **Architecture:** Consistent hashing (ring), tunable consistency (quorum), compaction strategies.
- **Use Case:** Time-series data, IoT sensor logs, Event logging, Messaging. *VD: Netflix, Apple, Instagram.*
- **Trade-offs:** Đọc chậm hơn (nhiều SSTable), không join, modeling phức tạp (query-first design).

Neo4j (Graph Database)

- **Mô hình:** Nodes (entities) + Relationships (edges) + Properties. Native graph storage.
- **Ưu điểm:** Traversal cực nhanh (follow pointers), query trực quan (Cypher), ACID transactions.
- **Architecture:** Index-free adjacency (mỗi node chứa pointer đến neighbors).
- **Use Case:** Social networks, Recommendation engines, Fraud detection, Knowledge graphs. *VD: LinkedIn, Walmart, eBay.*
- **Trade-offs:** Scale khó hơn NoSQL khác, không tốt cho bulk data processing.

3.7 Xử lý Batch (Batch Processing)

3.7.1 Apache Spark (Unified Analytics Engine)

Thay thế MapReduce với tốc độ cao hơn 100x (in-memory).

- **Core Concept:** RDD (Resilient Distributed Dataset) - immutable, partitioned, parallel.
  - *Transformations:* Lazy (map, filter, join) - tạo DAG.
  - *Actions:* Eager (collect, count, save) - trigger execution.
- **Components:**
  - *Spark SQL:* Query structured data (DataFrame/Dataset API).
  - *Spark Streaming:* Micro-batch streaming (DStream).
  - *MLlib:* Machine learning library (classification, clustering, etc.).
  - *GraphX:* Graph processing (PageRank, connected components).
- **Ưu điểm:** In-memory caching, lazy evaluation, DAG optimization, unified API (batch + streaming).
- **Nhược điểm:** Tốn RAM, không true streaming (micro-batch), overhead cho job nhỏ.
- **Use Case:** ETL, ML training, interactive analytics, log processing. *VD: Netflix, Uber, Airbnb.*

3.8 Xử lý Luồng (Streaming Processing)

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

3.8.1 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).

3.9 Giao thức IoT (IoT Protocols)

3.9.1 MQTT vs HTTP

HTTP (HyperText Transfer)	MQTT (Message Queuing)
Mô hình: Request - Response (Client-Server).	**Mô hình:** Publish - Subscribe (qua Broker).
Kết nối: Ngắn, đóng sau khi xong.	**Kết nối:** Dài, *Keep-alive*, nhẹ.
Header: Lớn, cồng kềnh (Metadata).	**Header:** Rất nhỏ (2 byte), tiết kiệm băng thông.
Use Case: Web, API, truyền tải lớn.	**Use Case:** IoT, mạng chập chờn, pin yếu.

3.10 Pipelines & Orchestration

3.10.1 Apache Kafka vs Airflow

Apache Kafka	Apache Airflow
Loại: Event Streaming Platform (Message Broker).	**Loại:** Workflow Orchestration (Quản lý quy trình).
Đặc trưng: • Log bền vững (Durable log). • Decoupling (Tách rời). • Replayable, High throughput.	**Đặc trưng:** • Code-as-infra (Python DAGs). • Quản lý dependency phức tạp. • Backfill (chạy lại quá khứ).
Vai trò: "Xương sống"vận chuyển dữ liệu *Real-time.*	**Vai trò:** "Nhạc trưởng"điều phối Job (*Batch/ETL*).

3.10.2 CDC (Change Data Capture)

Theo dõi và đồng bộ thay đổi từ database nguồn sang đích.

- **Mục đích:** Real-time data replication, sync giữa OLTP và OLAP, event-driven architecture.
- **Cơ chế:**
  - *Log-based CDC:* Đọc database transaction log (binlog MySQL, WAL PostgreSQL). **Tốt nhất** - không ảnh hưởng source.
  - *Trigger-based:* Trigger trên INSERT/UPDATE/DELETE. Ảnh hưởng performance.
  - *Timestamp/Version-based:* Poll dựa trên `updated_at` column. Thiếu DELETE events.
- **Tools:** Debezium (Kafka Connect), AWS DMS, Oracle GoldenGate, Airbyte.
- **Use Case:** Sync OLTP → DW, Microservices data sharing, Cache invalidation, Audit logs.
- **Pattern:** Source DB → CDC Tool → Kafka → Sink (DW/Cache/Search).

3.10.3 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).

3.11 Kho Dữ liệu (Data Warehousing)

3.11.1 OLTP vs OLAP

	OLTP (Transactional)	OLAP (Analytical)
Mục tiêu	Vận hành hàng ngày (Operational).	Ra quyết định (Decision support).
Dữ liệu	Hiện hành, chi tiết, cập nhật liên tục.	Lịch sử, tổng hợp, đa chiều.
Truy vấn	Đơn giản, trả về ít dòng (Lookup).	Phức tạp, join nhiều, quét bảng lớn.
Thiết kế	Chuẩn hóa cao (3NF) để tránh dị thường.	Phi chuẩn hóa (Star/Snowflake) để đọc nhanh.
User	NV, App, Khách hàng.	Manager, Data Analyst.

3.11.2 4 Đặc trưng Chính (Inmon)

1. **Hướng chủ đề (Subject-oriented):** Tổ chức theo chủ đề chính (Khách hàng, Sản phẩm) thay vì theo ứng dụng (App Bán hàng, App Kho).
2. **Tích hợp (Integrated):** Dữ liệu từ nhiều nguồn được làm sạch, đồng nhất (đơn vị, format, encoding) trước khi nạp.
3. **Bất biến (Non-volatile):** Dữ liệu đã vào DW thì (thường) không bị sửa/xóa, chỉ đọc.
4. **Biến thiên theo thời gian (Time-variant):** Mọi dữ liệu đều gắn với mốc thời gian để phân tích xu hướng (Historical data).

3.11.3 Thách thức Xây dựng DW

- **Data Quality:** "Garbage In, Garbage Out". Dữ liệu nguồn bẩn làm sai lệch báo cáo.
- **ETL Complexity:** Tích hợp các hệ thống cũ (Legacy) rất phức tạp.
- **Performance:** Truy vấn phân tích tốn tài nguyên, cần tối ưu index/partition.
- **User Acceptance:** Người dùng không hiểu hoặc không tin tưởng dữ liệu.
- **Cost:** Chi phí lưu trữ và duy trì hạ tầng cao.

3.11.4 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.*

3.11.5 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).

3.11.6 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.*

Type	Chiến lược	Đặc điểm & Use Case
0	**Retain Original:** Giữ nguyên, không bao giờ sửa.	Dữ liệu gốc là chân lý. (VD: Ngày sinh).
1	**Overwrite:** Ghi đè giá trị mới lên cũ.	Không cần lịch sử. Sửa lỗi chính tả.
2	**Add Row:** Thêm dòng mới + *Effective Date* + *Current_Flag*.	**Chuẩn mực nhất**. Theo dõi toàn bộ lịch sử biến động.
3	**Add Column:** Thêm cột *Previous_Value*.	Chỉ cần biết giá trị liền trước. (Ít dùng).
4	**Add History Table:** Tách bảng lịch sử riêng (Mini-Dimension).	Tối ưu khi bảng chính quá lớn và chỉ một nhóm thuộc tính thay đổi nhanh.
5	**Hybrid (4 + 1):** Mini-dimension + tham chiếu "Current"ở bảng chính.	Tối ưu truy vấn khi cần cả lịch sử chi tiết và giá trị hiện tại nhanh chóng.
6	**Hybrid (1 + 2 + 3):** Type 2 row + cột chứa giá trị hiện tại (Type 1).	*"Pure Type 6"*: Giúp truy vấn lịch sử nhưng vẫn group by theo giá trị hiện tại dễ dàng.
7	**Hybrid (Dual Keys):** Fact table chứa cả *Surrogate Key* (lịch sử) và *Natural Key* (hiện tại).	Linh hoạt nhất: Join theo Surrogate để xem lịch sử, join theo Natural để xem hiện tại.

Các kỹ thuật xử lý dữ liệu thay đổi theo thời gian (0 → 7).

3.12 Quản lý Dữ liệu

3.12.1 Vấn đề Tích hợp (Data Integration Issues)

- **Heterogeneous data sources:** Khác biệt về DBMSs và files.
- **Data Conflicts:** Khác kiểu dữ liệu (String vs Int, Date format).
- **Data Mapping:** Khác biệt về schema (vd: 'Full Name' vs 'First Name' + 'Last Name').
- **Entity Resolution:** Xác định 2 bản ghi từ 2 nguồn là cùng 1 thực thể (vd: 'Nguyen V. A' và 'A Nguyen').
- **Data redundancy:** Dữ liệu trùng lặp cần khử trùng (Deduplication).

3.12.2 6 Chiều Chất lượng Dữ liệu (Data Quality)

1. **Completeness (Đầy đủ):** Không bị thiếu (Null/Missing values).
2. **Accuracy (Chính xác):** Phản ánh đúng thực tế.
3. **Consistency (Nhất quán):** Dữ liệu thống nhất giữa các bảng/hệ thống.
4. **Validity (Hợp lệ):** Đúng định dạng, nằm trong miền giá trị cho phép.
5. **Timeliness (Kịp thời):** Dữ liệu có sẵn khi cần (độ trễ thấp).
6. **Uniqueness (Duy nhất):** Không trùng lặp.

3.12.3 Quản lý Thông tin (Information Management)

Là việc quản lý dữ liệu như một tài sản doanh nghiệp.

- Bao gồm: Data Governance (Quản trị), Data Quality, Master Data Management (MDM), Security.
- Mục tiêu: Đảm bảo dữ liệu tin cậy, an toàn, dễ truy cập để tạo giá trị.

3.13 Thách thức Xử lý Phân tán (Distributed Challenges)

- **Data Skew (Lệch dữ liệu):** Tình trạng một partition chứa quá nhiều dữ liệu so với các partition khác.
  → *Hậu quả:* "Straggler problem Cả job phải đợi node chậm nhất hoàn thành.
  → *Giải pháp:* Salting (thêm tiền tố ngẫu nhiên vào khóa) để chia nhỏ key hot.
- **Shuffle:** Quá trình chuyển dữ liệu giữa các node qua mạng (Map → Reduce).
  → *Tối ưu:* Dùng **Broadcast Join** (chép bảng nhỏ đến tất cả node chứa bảng lớn) để tránh shuffle bảng lớn.