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

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

Ngày 30 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

- **Phụ thuộc Hàm (FD):** $X \to Y$ nghĩa là các bộ giống nhau trên X thì phải giống nhau trên Y.
- **Các tính chất Thiết kế:**
  1. **Kết nối bảo toàn thông tin (Không tổn thất):** Phân rã không được tạo ra các bộ giả (spurious tuples).
  2. **Bảo toàn phụ thuộc:** Các FD ban đầu có thể kiểm tra được trên các quan hệ đã phân rã.
  3. **Giảm thiểu dư thừa:** Tránh các dị thường cập nhật (thêm, xóa, sửa).
- **Các Dạng chuẩn:** 1NF (nguyên tố), 2NF (FD đầy đủ vào khóa), 3NF (không có FD bắc cầu), BCNF (với mọi FD $X \to A$, X là siêu khóa; có thể không bảo toàn phụ thuộc). 4NF/5NF giải quyết phụ thuộc đa trị và phụ thuộc kết nối.
- **Phi chuẩn hóa (Denormalization):** Lưu trữ kết quả kết nối như quan hệ cơ sở để tăng hiệu năng nhưng chấp nhận dị thường.

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.

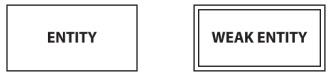
1.5 Các bước vẽ ERD (Steps to Create ERD)

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, KHACHHANG); **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).



(Thực thể)

(Thực thể yếu)

(Thuộc tính)

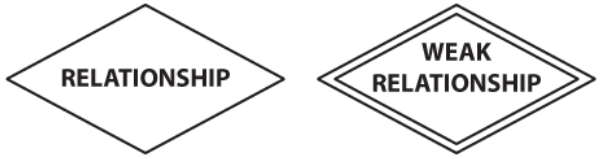
(Khóa)

(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, KHACHHANG *mua* SANPHAM); **Quan hệ xác định** khi thực thể yếu phụ thuộc vào thực thể chủ (NGUOIPHUOCTHUOC *thuộc về* NHANVIEN với khóa bộ phận là tên người phụ thuộc).



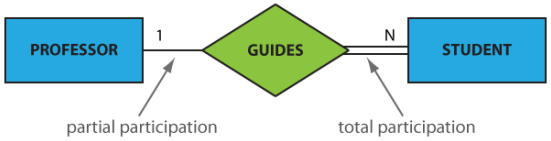
Bao gồm: Quan hệ (Hình thoi), Quan hệ xác định (Thoi đôi)

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):

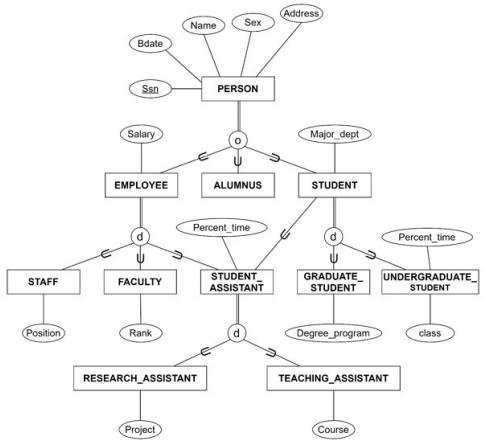


(Partial Participation (Từng phần): Nét đơn
Total Participation (Toàn phần): Nét đôi)

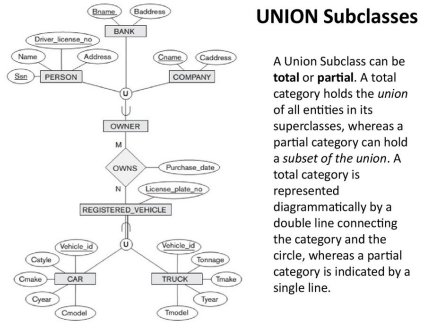
Min-Max (min, max): Ghi cặp số trên cạnh. Ví dụ: NHANVIEN $(1,1)$ *làm việc cho* $(0,N)$ PHONGBAN nghĩa là mỗi nhân viên bắt buộc làm việc cho đúng 1 phòng ban, mỗi phòng ban có thể có 0 đến nhiều nhân viên. $(0,1)$: tùy chọn, tối đa 1; $(1,N)$: bắt buộc, có thể nhiều. (An extension of Participation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

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:** $15 + 240 + 3{,}840 \approx \mathbf{4{,}095}$ 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: $12{,}167 \times 21 \approx \mathbf{255{,}507}$ 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: $\sigma_{\text{Ssn}='123456789'}$(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: $\sigma_{\text{Dno}=5}$(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: $\sigma_{\text{Dno}=5 \wedge \text{Salary}>30\,000 \wedge \text{Sex}='F'}$(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

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 $\rightarrow$ EMPLOYEE trong:** Mỗi lần tìm $= x + s = 2 + 80 = 82$. Tổng $= 13 + 125 \times 82 + 2{,}500 = \mathbf{12{,}763}$.
- **EMPLOYEE ngoài $\rightarrow$ 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 $\Rightarrow$ 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 Lưu trữ Dữ liệu Lớn

3.1 Khái niệm

- **Động lực NoSQL:** Khả năng mở rộng ngang, tính sẵn sàng, lược đồ linh hoạt (BASE/CAP) so với ACID chặt chẽ.
- **3V của Big Data:** Volume (TB–PB–EB), Velocity (thời gian thực), Variety (cấu trúc/bán cấu trúc/phi cấu trúc); thường thêm Veracity (độ tin cậy) và Value (giá trị).
- **Chiến lược lược đồ:** *Lược đồ khi ghi (Schema-on-write)* (DW) so với *Lược đồ khi đọc (Schema-on-read)* (DL).
- **Lưu trữ phân tán:** Hệ thống kiểu HDFS cung cấp lưu trữ chịu lỗi, mở rộng trên các cụm máy.

3.1.1 Động lực NoSQL (BASE/CAP vs ACID)

- **ACID vs BASE:** RDBMS tuân thủ ACID; NoSQL thường theo BASE (Basically Available, Soft state, Eventually consistent) để mở rộng.
- **Đánh đổi CAP:** Trong hệ thống phân tán có sao chép, không thể đồng thời có Consistency (Nhất quán), Availability (Sẵn sàng), và Partition tolerance (Chịu phân hoạch). NoSQL thường chọn AP hơn C chặt chẽ.
- **Nhất quán cuối cùng (Eventual consistency):** Các bản sao có thể lệch nhau tạm thời nhưng sẽ hội tụ nếu không có cập nhật mới; chấp nhận được cho nhiều ứng dụng web.
- **Mở rộng ngang (Scale-out):** Mở rộng qua phân mảnh (sharding) và sao chép (replication) trên các node phổ thông.
- **Lược đồ linh hoạt:** Bản ghi tự mô tả (JSON/BSON), bán cấu trúc, thuộc tính không đồng nhất.
- **Mô hình:** Văn bản (MongoDB), Khóa-Giá trị (Redis/DynamoDB), Cột rộng (BigTable/HBase/Cassandra), Đồ thị (Neo4j).

3.1.2 5 chữ V của Big Data

- **Volume (Dung lượng):** TB đến PB đến EB; từ log, di động, giao dịch, cảm biến IoT; cần song song hóa lớn.
- **Velocity (Tốc độ):** Tốc độ nạp cao và xử lý luồng/thời gian thực để phát hiện gian lận, giám sát.
- **Variety (Đa dạng):** Cấu trúc, bán cấu trúc, phi cấu trúc (web, mạng xã hội, vị trí, ảnh, video, log). Dữ liệu phi cấu trúc là thách thức lớn.
- **Veracity (Độ xác thực):** Độ tin cậy/chất lượng dữ liệu biến thiên; cần kiểm tra trước khi phân tích.
- **Value (Giá trị):** Phân tích (mô tả/dự đoán/đề xuất) để tạo ra lợi ích kinh doanh.

3.1.3 Lưu trữ Phân tán (Kiểu HDFS)

- **Không chia sẻ (Shared-nothing):** Mỗi node có CPU/RAM/đĩa riêng; phối hợp qua mạng; mở rộng kinh tế nhờ dư thừa.
- **Kiến trúc HDFS:** NameNode quản lý không gian tên/metadata; DataNode lưu khối và phục vụ I/O; cụm có thể có hàng ngàn DataNode.
- **Sao chép (Replication):** Khối được sao chép (thường 3x) để bền vững và sẵn sàng; client đọc từ bản sao gần nhất.
- **I/O Song song:** Nhiều máy đọc/ghi đồng thời, tăng tổng thông lượng.
- **Chỉ thêm (Append-only):** Mô hình nhất quán đơn giản tối ưu cho batch: tập tin chỉ được thêm vào cuối (không update ngẫu nhiên).
- **Hệ sinh thái:** HDFS làm nền tảng cho MapReduce, HBase, và các công cụ big data khác.

3.2 Các Công nghệ

Document Store (MongoDB)

- *Lược đồ khi đọc* linh hoạt cho tính Đa dạng (Variety); thuộc tính không đồng nhất.
- Tài liệu JSON/BSON (mảng, đối tượng lồng); thiết kế phi chuẩn hóa để tăng tính cục bộ dữ liệu.
- CRUD qua `find(<cond>)`; tự động đánh chỉ mục _id để truy xuất theo khóa.
- Tính sẵn sàng cao qua Replica Sets (đọc primary/secondary).
- Mở rộng ngang với Sharding trên khóa shard (khoảng/băm; query router).
- MapReduce chỉ đọc trên các tập hợp (collection) lớn.

Key-Value Cache (Redis)

- Hash-map trong bộ nhớ (In-memory) cho tốc độ đọc cực nhanh.
- Hỗ trợ cấu trúc dữ liệu phức tạp (set, queue) một cách tự nhiên.
- Bền vững qua nhật ký append-only và snapshot.
- Mẫu Read-through cache; ứng dụng quản lý việc vô hiệu hóa cache.
- Sao chép Master-slave cho tính sẵn sàng cao.

Graph DB (Neo4j)

- Mô hình đồ thị thuộc tính: nút, quan hệ, nhãn, thuộc tính.
- Truy vấn đường dẫn hiệu quả; duyệt độ dài thay đổi qua Cypher.
- Ngôn ngữ khai báo với ký hiệu mũi tên cho các mẫu (patterns).
- Tính năng doanh nghiệp: caching, clustering, sao chép master-slave.
- Thiết kế tập trung tối ưu cho tải công việc đồ thị.

Wide-Column Stores (BigTable/HBase/Cassandra)

- Bản đồ đa chiều thưa được sắp xếp: row key, column info, phiên bản.
- Lưu trữ LSM-tree chuyển đổi ghi ngẫu nhiên thành I/O tuần tự.
- HBase trên HDFS; ZooKeeper để điều phối; vùng (region) theo khoảng khóa.
- Column families nhóm lưu trữ; bộ định danh cột (qualifier) định nghĩa động.
- Cassandra: Sao chép không leader kiểu Dynamo; băm nhất quán (consistent hashing).
- Chiến lược nén (vd: cửa sổ thời gian); chỉ mục phụ SASI.

Data Warehouse Engines

- HiveQL trên Hadoop; biên dịch sang kế hoạch thực thi MapReduce/Tez/Spark.

– SerDe hiển thị tập tin thô dưới dạng bảng; đẩy vị từ (predicate pushdown).
– Định dạng cột ORC/Parquet giảm thiểu I/O cho phân tích.
– Cloud OLAP: Snowflake/BigQuery/Redshift cho phân tích tương tác.
– Thực thi song song với lưu trữ mở rộng.

Khung xử lý (Processing Frameworks)

– MapReduce: map/shuffle/reduce trên HDFS/HBase; chịu lỗi thông qua thực thi lại.
– Spark: RDD trong bộ nhớ; động cơ thống nhất (SQL, đồ thị, ML, streaming).
– Spark trên YARN; đọc từ HDFS/HBase.
– Tez: Thực thi DAG; pipeline các giai đoạn để tránh hiện thực hóa xuống HDFS.
– Nền tảng cho các hệ thống cấp cao hơn như Hive.

3.3 Xử lý Luồng (Streaming)

– **Mục đích:** Xử lý thời gian thực, không giới hạn cho phát hiện gian lận/xâm nhập, theo dõi, giám sát.

Thông điệp: Kafka Durable Log & CDC

– **Lưu trữ Log:** Đĩa append-only; producer thêm vào, consumer đọc tuần tự.
– **Phân hoạch/Thứ tự:** Phân hoạch Topic với thứ tự toàn cục trên mỗi phân hoạch qua offset.
– **Độ bền/Replay:** Retention giữ lại các bộ; việc tiêu thụ là chỉ đọc; consumer có thể replay từ offset.
– **CDC transport:** Luồng Change Data Capture bảo toàn thứ tự; DB nguồn là leader, follower xây lại trạng thái.
– **Nén Log:** Giữ lại lần ghi cuối cùng cho mỗi khóa, cho phép snapshot bảng mới nhất.

Tính toán: Flink/Spark & Cloud

– **Apache Flink:** Động cơ streaming thực thụ; thực thi pipeline; checkpoint để phục hồi.
– **Spark Streaming:** Microbatching (1s) trên RDD; tính toán lại khi lỗi.
– **Cloud:** Kinesis (log-based), Dataflow (checkpointed pipelines), Azure Stream Analytics (SQL streaming được quản lý).

Cửa sổ (Windows): Tumbling vs Hopping

– **Tumbling (Trượt không chồng):** Độ dài cố định, liền kề, không chồng lấp (vd: 1 phút).
– **Hopping (Trượt chồng lấp):** Độ dài cố định, có chồng lấp (vd: cửa sổ 5 phút trượt mỗi 1 phút).
– **Ngữ nghĩa thời gian:** Thời gian sự kiện (Event time) vs Thời gian xử lý (Processing time); quản lý sự kiện đến muộn.

Stream Joins: Có giới hạn thời gian & Có trạng thái

– **Stream-Stream:** Kết nối cửa sổ trên khóa có giới hạn thời gian (vd: trong vòng 30 phút).
– **Stream-Table:** Làm giàu sự kiện dùng một quan hệ/changelog được xem như bảng.
– **Table-Table:** Kết nối các changelog để duy trì khung nhìn vật lý hóa (vd: tweets × follows).
– **Trạng thái (State):** Bộ xử lý duy trì trạng thái theo khóa và bộ đệm cửa sổ để khớp các sự kiện đến.

Ngữ nghĩa Exactly-Once

– **Mục tiêu:** Đầu ra tương đương với việc thực thi không lỗi (không mất/không trùng).
– **Phục hồi Framework:** Microbatching (Spark) & checkpointing (Flink) khởi động lại từ điểm nhất quán; loại bỏ đầu ra một phần.
– **Giao dịch phân tán:** Atomic commits cho trạng thái + thông điệp trong bộ xử lý.
– **Tính lũy đẳng bên ngoài:** Dùng offset/khóa duy nhất, bền vững để làm các tác dụng phụ trở nên lũy đẳng (khử trùng khi ghi).

3.4 Pipelines: ETL vs ELT

– **ETL:** Extract → Transform → Load (biến đổi trước khi nạp).
– **ELT:** Extract → Load → Transform (dùng khả năng tính toán của kho dữ liệu; schema-on-read).
– **Kafka:** Transport/CDC stream cho Extract/Load.
– **Airflow:** Điều phối các phụ thuộc batch cho Transform/Load.

3.5 IoT

– **Nguồn:** Cảm biến/RFID (tốc độ cao, dữ liệu tín hiệu đa dạng).
– **Ứng dụng:** Nông nghiệp thông minh, giám sát vận động viên, theo dõi di chuyển.
– **Giao thức:** MQTT cho nạp dữ liệu thời gian thực.

3.6 So sánh

MongoDB vs Redis vs Neo4j

– **MongoDB:** Document store, BSON, replica sets, sharding, MapReduce.
– **Redis:** In-memory key-value/cache, bền vững, sao chép; đọc nhanh.
– **Neo4j:** Mô hình đồ thị, Cypher cho đường dẫn, clustering; truy vấn đồ thị hiệu quả.

Cassandra vs Hive vs Snowflake

– **Cassandra:** Wide-column, sao chép không leader, nhất quán cuối cùng, LSM + compaction, SASI.
– **Hive:** SQL-on-Hadoop biên dịch sang MapReduce/Tez/Spark; SerDe cho ORC/Parquet.
– **Snowflake:** Cloud DW OLAP; tương đương BigQuery/Redshift.

BigTable vs BigQuery

– **BigTable:** Lưu trữ cột rộng phân tán; bản đồ thưa được sắp xếp; truy vấn khoảng; cảm hứng cho HBase.
– **BigQuery:** Phân tích OLAP tương tác; dòng dõi Dremel; động cơ SQL ở quy mô web.

3.7 Kho dữ liệu (Data Warehousing)

– **Đặc điểm:** Hướng chủ đề, tích hợp, bất biến, biến thiên theo thời gian; ít truy vấn nhưng quét lớn.
– **Lược đồ:** Bảng Fact (độ đo + khóa chiều) và bảng Dimension (chiều); Hình sao (Star) vs Bông tuyết (Snowflake).
– **Lưu trữ:** Ưu tiên hướng cột; Teradata, Sybase IQ, Redshift; Oracle/HANA/SQL Server hỗ trợ cột.
– **Tích hợp Big Data:** Hadoop với Hive/Spark SQL cho SQL trên tập tin phân tán.

3.8 Mô hình Đa chiều

– **Dạng bảng (Dimensional):** Lược đồ Star/Snowflake cho phép phân tích chiều.
– **Data Cube:** `GROUP BY CUBE` (mọi tập con); `ROLLUP` (tập con phân cấp).

3.9 DW vs DL vs Lakehouse

– **DW (Kho dữ liệu):** Schema-on-write; nhất quán mạnh; dữ liệu có cấu trúc; OLAP (ETL).
– **DL (Hồ dữ liệu):** Schema-on-read; dữ liệu thô đa định dạng; lưu trữ rẻ; Hadoop/Spark để truy vấn.
– **Lakehouse:** Lai ghép, kết hợp sự linh hoạt của hồ với quản lý của kho (ACID, thực thi lược đồ, đánh chỉ mục).