### **\*\* slide 1: tiêu đề\*\***

**SO SÁNH ORACLE VÀ POSTGRESQL** **Một Cái Nhìn Toàn Diện về Hiệu Năng, Tính Năng và Chi Phí**

### **\*\* slide 2: mục tiêu\*\***

**(0:00:00) 0. Mục tiêu**

* **Mục tiêu chính:** Cung cấp một cái nhìn tổng quan và sâu sắc về những điểm khác biệt cốt lõi giữa hai hệ quản trị cơ sở dữ liệu (CSDL) hàng đầu: Oracle Database và PostgreSQL.
* **Đối tượng:** Các nhà phát triển, quản trị viên CSDL (DBA), kiến trúc sư hệ thống và các nhà quản lý kỹ thuật.
* **Kết quả mong đợi:** Giúp người nghe có đủ thông tin để đưa ra lựa chọn phù hợp cho dự án của mình dựa trên các yếu\_tố về kỹ thuật, vận hành và chi phí.

### **\*\* slide 3: hiệu năng giao dịch đồng thời - lý thuyết chung\*\***

**(0:01:40) 1. Hiệu năng khi có nhiều giao dịch đồng thời**

Cả Oracle và PostgreSQL đều sử dụng cơ chế **MVCC (Multi-Version Concurrency Control)** để xử lý các giao dịch đồng thời. Cơ chế này cho phép người đọc (câu lệnh SELECT) không bị chặn bởi người viết (câu lệnh INSERT, UPDATE, DELETE) và ngược lại.

Tuy nhiên, cách triển khai MVCC của chúng hoàn toàn khác nhau, dẫn đến sự khác biệt lớn về hiệu năng và cách vận hành.

### **\*\* slide 4: cơ chế xử lý giao dịch đồng thời\*\***

**(0:01:45) 1.1 Cơ chế xử lý giao dịch đồng thời Oracle và PostgreSQL**

#### **Lý thuyết**

* **Oracle: Mô hình "UNDO" Segment**
  + Khi một dòng dữ liệu bị thay đổi, Oracle sẽ ghi phiên bản **mới** của dữ liệu trực tiếp vào block dữ liệu (data block).
  + Phiên bản **cũ** của dữ liệu sẽ được sao chép vào một khu vực riêng gọi là **UNDO Segment**.
  + Khi một giao dịch khác cần đọc dữ liệu tại thời điểm trước khi thay đổi, Oracle sẽ tìm đến UNDO Segment để tái tạo lại phiên bản dữ liệu đó.
  + **Ưu điểm:** Tách biệt dữ liệu cũ và mới, không làm "phình" bảng dữ liệu chính. Việc dọn dẹp (quản lý UNDO) diễn ra tự động và hiệu quả.
* **PostgreSQL: Mô hình "Tuple" mới**
  + Khi một dòng dữ liệu được UPDATE, PostgreSQL không ghi đè lên dữ liệu cũ. Thay vào đó, nó tạo ra một phiên bản **mới** của dòng đó (gọi là "new tuple") và đánh dấu phiên bản **cũ** là "hết hạn" (expired). Cả hai phiên bản cùng tồn tại trong bảng.
  + Các giao dịch đang chạy sẽ "nhìn thấy" phiên bản dữ liệu phù hợp với "snapshot" của chúng.
  + Một tiến trình nền gọi là **VACUUM** có nhiệm vụ dọn dẹp các phiên bản cũ đã "chết" (không còn giao dịch nào cần đến) để thu hồi không gian.
  + **Nhược điểm:** Có thể gây ra hiện tượng "table bloat" (bảng bị phình to) nếu VACUUM không được cấu hình và chạy thường xuyên, hiệu quả.

#### **Ví dụ minh họa dễ hiểu**

Hãy tưởng tượng bạn đang sửa một tài liệu Word.

* **Cách của Oracle:** Bạn sửa trực tiếp vào tài liệu. Trước khi sửa, hệ thống tự động copy đoạn văn bản cũ vào một "Sổ nháp" (UNDO Segment). Ai đó muốn xem phiên bản gốc thì nhìn vào "Sổ nháp".
* **Cách của PostgreSQL:** Bạn không sửa file cũ. Thay vào đó, bạn copy toàn bộ nội dung, dán vào cuối tài liệu, rồi sửa trên bản copy đó. Phiên bản cũ vẫn còn đó nhưng được đánh dấu "không dùng nữa". Sau một thời gian, một người "dọn dẹp" (VACUUM) sẽ đến và xóa những phiên bản cũ đi.

### **\*\* slide 5: leo thang lock - cực kỳ quan trọng\*\***

**(0:15:59) 1.2 Hiện tượng leo thang Lock (Lock Escalation) - Đặc biệt quan trọng**

#### **Lý thuyết**

Leo thang lock là hiện tượng hệ quản trị CSDL tự động chuyển đổi nhiều khóa cấp thấp (như khóa dòng - row lock) thành một khóa cấp cao hơn (như khóa toàn bộ bảng - table lock) để tiết kiệm tài nguyên bộ nhớ quản lý khóa.

* **Oracle: KHÔNG CÓ LEO THANG LOCK**
  + Oracle nổi tiếng với khả năng khóa ở mức độ dòng (row-level locking) và **không bao giờ** tự động leo thang lên khóa bảng.
  + Một giao dịch có thể khóa hàng triệu dòng riêng lẻ mà không ảnh hưởng đến các dòng khác trong cùng một bảng.
  + **Hệ quả:** Rất phù hợp cho các hệ thống OLTP (Online Transaction Processing) có lượng giao dịch cực lớn, độ tương tranh cao. Người dùng gần như không bao giờ bị chặn một cách không cần thiết.
* **PostgreSQL: CÓ LEO THANG LOCK**
  + PostgreSQL sử dụng một vùng nhớ có kích thước cố định để quản lý các khóa.
  + Khi một giao dịch thực hiện quá nhiều khóa dòng và sắp làm đầy vùng nhớ này (vượt ngưỡng max\_locks\_per\_transaction), PostgreSQL sẽ tự động **leo thang**, thay thế tất cả các khóa dòng bằng một khóa duy nhất trên toàn bộ bảng.
  + **Hệ quả:** Có thể gây ra vấn đề hiệu năng nghiêm trọng. Một tác vụ cập nhật hàng loạt (batch update) vô tình có thể khóa cả bảng, khiến tất cả các giao-dịch khác (kể cả SELECT) muốn truy cập vào bảng đó phải chờ đợi, gây "treo" ứng dụng.

#### **Ví dụ minh họa dễ hiểu**

Bạn có một bãi giữ xe với 1000 chỗ.

* **Cách của Oracle:** Bạn có một nhân viên siêu năng lực có thể nhớ chính xác vị trí của từng chiếc xe trong 1000 chỗ đó. Dù có 900 xe vào lấy khóa, anh ta vẫn xử lý từng xe một cách độc lập.
* **Cách của PostgreSQL:** Bạn có một nhân viên với một cuốn sổ chỉ ghi được 100 dòng. Khi có hơn 100 xe vào gửi, để không bị quá tải sổ sách, anh ta quyết định **đóng cổng** toàn bộ bãi xe và treo biển "Bãi xe đang được quản lý, vui lòng không ra vào". Không ai có thể vào hay ra cho đến khi công việc của anh ta xong.

### **\*\* slide 6: yếu tố ảnh hưởng hiệu năng sql - partition\*\***

**(0:23:20) 1.3 Yếu tố ảnh hưởng hiệu năng SQL / PARTITION**

#### **Lý thuyết**

Cả hai đều hỗ trợ Partitioning (phân vùng), một kỹ thuật chia một bảng lớn thành nhiều mảnh nhỏ hơn (partition) nhưng vẫn được truy vấn như một bảng duy nhất. Việc này giúp cải thiện hiệu năng truy vấn (chỉ quét các partition cần thiết) và dễ dàng quản lý dữ liệu (ví dụ xóa dữ liệu cũ bằng cách drop partition).

* **Oracle:** Rất mạnh mẽ và trưởng thành. Hỗ trợ nhiều kiểu phân vùng phức tạp (Range, List, Hash, Composite). Tính năng này là một **tùy chọn trả phí** (Partitioning Option) trong phiên bản Enterprise Edition.
* **PostgreSQL:** Hỗ trợ đầy đủ phân vùng (Range, List, Hash) từ phiên bản 10 và ngày càng được cải thiện. Đây là tính năng **miễn phí** có sẵn.

#### **Ví dụ minh họa (Cú pháp)**

**Mục tiêu:** Phân vùng bảng sales theo tháng.

* **Oracle:**
* SQL

### **\*\* slide 7: yếu tố ảnh hưởng hiệu năng sql - index\*\***

**(0:29:39) 1.3 Yếu tố ảnh hưởng hiệu năng SQL / INDEX**

#### **Lý thuyết**

Cả hai đều cung cấp các loại index tiêu chuẩn như B-Tree. Tuy nhiên, PostgreSQL nổi bật với hệ thống index có khả năng mở rộng và nhiều loại index chuyên dụng.

* **Oracle:** Có các loại index rất mạnh mẽ như B-Tree, Bitmap, Function-Based Index.
* **PostgreSQL:** Ngoài B-Tree, PostgreSQL cung cấp các loại index độc đáo và mạnh mẽ, là một phần của mã nguồn mở:
  + **GIN (Generalized Inverted Index):** Cực kỳ hiệu quả cho việc tìm kiếm trên các kiểu dữ liệu phức hợp như mảng (array) hoặc dữ liệu text (full-text search).
  + **GiST (Generalized Search Tree):** Hữu ích cho việc index dữ liệu hình học, địa lý (PostGIS).
  + **BRIN (Block Range Indexes):** Hiệu quả cho các bảng rất lớn có dữ liệu được sắp xếp tự nhiên (ví dụ: bảng log theo thời gian).

#### **Ví dụ minh họa**

**Mục tiêu:** Tìm kiếm tất cả sản phẩm có tag là 'khuyến mãi'.

Giả sử bảng products có một cột tags kiểu TEXT[] (mảng các text).

* **Cách tiếp cận thông thường (chậm):**
* SQL

### **\* slide 8: yếu tố ảnh hưởng hiệu năng sql - parallel\*\***

**(0:30:52) 1.3 Yếu tố ảnh hưởng hiệu năng SQL / PARALLEL**

#### **Lý thuyết**

Cả hai CSDL đều có khả năng thực thi song song (Parallel Execution), tức là sử dụng nhiều nhân CPU để xử lý một câu lệnh SQL duy nhất, giúp tăng tốc các tác vụ nặng như quét bảng lớn, join, tổng hợp dữ liệu.

* **Oracle:**
  + Cơ chế Parallel Execution rất trưởng thành, mạnh mẽ và có thể được điều khiển chi tiết thông qua các "hint" trong câu lệnh SQL hoặc các chính sách (policy) trên bảng.
  + DBA có toàn quyền kiểm soát mức độ song song.
* **PostgreSQL:**
  + Tính năng thực thi song song đã được cải tiến vượt bậc trong các phiên bản gần đây (từ bản 10 trở đi).
  + Bộ tối ưu hóa (Optimizer) của PostgreSQL sẽ tự động quyết định khi nào nên sử dụng truy vấn song song dựa trên chi phí (cost). Người dùng có thể ảnh hưởng đến quyết định này bằng cách điều chỉnh các tham số hệ thống (max\_parallel\_workers\_per\_gather).

#### **Ví dụ minh họa**

**Mục tiêu:** Đếm số lượng bản ghi trong một bảng rất lớn.

* **Oracle (Dùng hint để yêu cầu chạy song song với 8 tiến trình):**
* SQL

### **\* slide 9: phân tích hiệu năng và xử lý sự cố - awr\*\***

**(0:34:17) 2. Sự ưu việt trong phân tích hiệu năng và xử lý sự cố** **(0:34:22) 2.1 AWR - Bắt buộc phải biết khi tối ưu Database Oracle**

#### **Lý thuyết**

* **Oracle: AWR (Automatic Workload Repository)**
  + Là công cụ chẩn đoán hiệu năng **tích hợp sẵn** và **tự động** của Oracle.
  + Nó tự động chụp lại các "snapshot" (ảnh chụp nhanh) về mọi khía cạnh hoạt động của CSDL theo chu kỳ (mặc định là 1 giờ).
  + AWR Report cung cấp một báo cáo cực kỳ chi tiết về:
    - Top SQL (theo thời gian CPU, số lần chạy, thời gian chờ...).
    - Các sự kiện chờ đợi (Wait Events) hàng đầu.
    - Thông số hệ điều hành và CSDL.
    - Hiệu quả của các thành phần bộ nhớ (Buffer Cache, Shared Pool...).
  + Đây là công cụ **vô giá** cho DBA để chẩn đoán các vấn đề hiệu năng "dai dẳng" hoặc đột xuất.
  + **Lưu ý:** AWR là một phần của **Diagnostics Pack**, một tùy chọn **trả phí** của phiên bản Enterprise.
* **PostgreSQL: Công cụ mã nguồn mở**
  + PostgreSQL không có một công cụ tương đương AWR được tích hợp sẵn.
  + Việc giám sát và chẩn đoán hiệu năng dựa vào sự kết hợp của nhiều công cụ:
    - **pg\_stat\_statements:** Một extension (tiện ích mở rộng) để theo dõi thống kê thực thi của các câu lệnh SQL. Phải được cài đặt và cấu hình.
    - **EXPLAIN (ANALYZE, BUFFERS):** Lệnh để xem kế hoạch thực thi chi tiết của một câu lệnh.
    - **Công cụ của bên thứ ba:** pgBadger (phân tích log), PMM (Percona Monitoring and Management), pg\_stat\_monitor, TimescaleDB... để trực quan hóa dữ liệu.

#### **Ví dụ minh họa**

* **Kịch bản Oracle:** "Hệ thống bị chậm vào khoảng 3-4 giờ chiều hôm qua".
  + **Hành động của DBA:**
    1. Chạy script tạo AWR report cho khoảng thời gian đó: @awrrpt.sql.
    2. Nhập begin\_snap và end\_snap tương ứng.
    3. Phân tích file HTML/text được tạo ra, tìm đến mục "SQL ordered by Elapsed Time" để xác định ngay thủ phạm.
* **Kịch bản PostgreSQL:** "Hệ thống bị chậm vào khoảng 3-4 giờ chiều hôm qua".
  + **Hành động của DBA:**
    1. Kiểm tra dashboard của công cụ giám sát (PMM, Grafana) xem có gì bất thường về CPU, I/O, số lượng kết nối không.
    2. Truy vấn view pg\_stat\_statements để xem các câu lệnh tốn nhiều total\_exec\_time nhất.
    3. Phân tích file log của PostgreSQL trong khoảng thời gian đó (có thể dùng pgBadger).
  + **Kết luận:** Việc chẩn đoán trên PostgreSQL đòi hỏi nhiều công cụ và kỹ năng kết hợp hơn, trong khi Oracle cung cấp một giải pháp "tất cả trong một".

### **\*\* slide 10: công nghệ flashback\*\***

**(0:42:36) 2.2 Công nghệ Flashback xử lý sự cố ngay tức thì**

#### **Lý thuyết**

Flashback là một tập hợp các công nghệ độc quyền của Oracle, cho phép "tua ngược" dữ liệu hoặc cả CSDL về một thời điểm trong quá khứ mà **không cần phục hồi từ backup**.

* **Oracle: Flashback Technology**
  + Cho phép thực hiện các thao tác mạnh mẽ như:
    1. **Flashback Query:** SELECT ... AS OF TIMESTAMP ... để xem dữ liệu của một bảng tại một thời điểm quá khứ.
    2. **Flashback Table:** FLASHBACK TABLE ... TO TIMESTAMP ... để đưa cả một bảng về trạng thái cũ, phục hồi sau lỗi xóa/cập nhật nhầm.
    3. **Flashback Drop:** Khôi phục một bảng vừa bị DROP.
    4. **Flashback Database:** Đưa cả CSDL về một thời điểm trong quá khứ, cực nhanh để phục hồi sau lỗi logic nghiêm trọng.
  + Hoạt động dựa trên dữ liệu trong UNDO Segment và Flashback Log.
* **PostgreSQL: Point-in-Time Recovery (PITR)**
  + PostgreSQL không có công nghệ tương tự Flashback.
  + Để quay về một thời điểm trong quá khứ, DBA phải thực hiện **PITR**:
    1. Khôi phục (restore) từ một bản backup vật lý gần nhất.
    2. Cấu hình để server chạy ở chế độ "recovery mode", đọc lại các file lưu trữ log giao dịch (WAL archive) cho đến đúng thời điểm mong muốn.
  + **So sánh:** PITR mạnh mẽ nhưng là một quá trình "offline" (CSDL không khả dụng trong lúc restore) và chậm hơn nhiều so với Flashback Table/Database.

#### **Ví dụ minh họa**

**Sự cố:** Một lập trình viên vô tình chạy lệnh UPDATE employees SET salary = 1000; mà không có mệnh đề WHERE. Toàn bộ lương nhân viên bị sai.

* **Hành động trên Oracle:**
  + DBA chỉ cần chạy một lệnh duy nhất. CSDL vẫn online.
* SQL

### **\* slide 11: tính sẵn sàng và bảo mật\*\***

**(0:48:10) 3. Tính sẵn sàng (High Availability) và Bảo mật**

#### **Lý thuyết**

* **Tính Sẵn Sàng (HA)**
  + **Oracle:** Cung cấp giải pháp HA hàng đầu thế giới là **RAC (Real Application Clusters)**.
    - **Kiến trúc:** Active-Active, nhiều server (instance) cùng truy cập vào một bộ lưu trữ (shared disk).
    - **Lợi ích:** Cân bằng tải (load balancing) và chuyển đổi dự phòng (failover) tức thì mà ứng dụng không hề hay biết.
    - Đây là giải pháp cực kỳ đắt đỏ và phức tạp.
  + **PostgreSQL:**
    - **Kiến trúc:** Dựa trên **Streaming Replication** (sao chép đồng bộ/bất đồng bộ).
    - **Mô hình:** Active-Passive (một server chính ghi/đọc, các server phụ chỉ đọc - Hot Standby).
    - **Failover:** Cần các công cụ của bên thứ ba như **Patroni**, **Stolon** để tự động hóa việc chuyển đổi khi server chính gặp sự cố.
    - **Kết luận:** Giải pháp của PostgreSQL rất mạnh mẽ, tin cậy và chi phí thấp, nhưng không "liền mạch" và "active-active" như Oracle RAC.
* **Bảo mật**
  + **Oracle:** Cung cấp các tính năng bảo mật nâng cao, thường là **tùy chọn trả phí**:
    - **Transparent Data Encryption (TDE):** Mã hóa toàn bộ dữ liệu trên đĩa một cách trong suốt.
    - **Database Vault:** Kiểm soát truy cập nghiêm ngặt, ngay cả DBA cũng không thể xem dữ liệu nhạy cảm.
    - **Label Security:** Phân quyền truy cập đến từng dòng dữ liệu.
  + **PostgreSQL:**
    - Bảo mật cốt lõi rất mạnh mẽ: Role-based access control, SSL, Row-Level Security (RLS).
    - Mã hóa dữ liệu có thể được thực hiện ở cấp độ cột bằng extension pgcrypto, hoặc mã hóa toàn bộ ổ đĩa ở cấp hệ điều hành.
    - **Kết luận:** PostgreSQL cung cấp các công cụ bảo mật mạnh mẽ và miễn phí. Oracle cung cấp các giải pháp bảo mật cấp doanh nghiệp, tuân thủ các tiêu chuẩn khắt khe nhất, nhưng với chi phí cao.

### **\*\* slide 12: yếu tố lịch sử và chiến lược phát triển\*\***

**(0:51:43) 4. Yếu tố lịch sử và chiến lược phát triển**

* **Oracle**
  + **Lịch sử:** Ra đời từ năm 1977, là một trong những CSDL quan hệ thương mại đầu tiên. Luôn tập trung vào thị trường doanh nghiệp lớn (enterprise).
  + **Chiến lược:**
    - **"Land and Expand":** Bán sản phẩm cốt lõi và sau đó bán thêm các tùy chọn, gói mở rộng (Packs, Options).
    - Phát triển theo mô hình **mã nguồn đóng**. Chu kỳ phát hành dài, tập trung vào sự ổn định.
    - Hệ sinh thái sản phẩm khổng lồ (Fusion Middleware, Applications, Cloud).
    - Gắn chặt khách hàng vào hệ sinh thái của mình.
* **PostgreSQL**
  + **Lịch sử:** Có nguồn gốc từ dự án Ingres tại Đại học California, Berkeley (1986). Trở thành mã nguồn mở từ năm 1996.
  + **Chiến lược:**
    - Được phát triển bởi một **cộng đồng toàn cầu**.
    - Tập trung vào việc tuân thủ chuẩn SQL, khả năng mở rộng (extensibility) và sự đúng đắn của dữ liệu.
    - Chu kỳ phát hành **hàng năm** (mỗi năm một phiên bản lớn), đổi mới nhanh chóng.
    - Được mệnh danh là **"The World's Most Advanced Open Source Relational Database"**.
    - Chiến lược phát triển minh bạch và hướng tới cộng đồng.

### **\*\* slide 13: so sánh license\*\***

**(0:58:43) 5. So sánh License Oracle vs PostgreSQL**

#### **Lý thuyết**

Đây là một trong những khác biệt lớn nhất và thường là yếu tố quyết định.

* **Oracle: Phức tạp và đắt đỏ**
  + **Mô hình:**
    - **Per Processor (Core-based):** Trả tiền theo số lượng core của CPU trên server chạy CSDL. Phải nhân với "core factor" của Oracle.
    - **Named User Plus (NUP):** Trả tiền theo số lượng người dùng được định danh.
  + **Chi phí ẩn:** Rất nhiều tính năng quan trọng (Partitioning, RAC, AWR, TDE...) không có sẵn trong bản Enterprise Edition tiêu chuẩn, mà phải mua thêm các **Option** và **Management Pack** với chi phí rất cao.
  + **Kiểm toán (Audit):** Oracle nổi tiếng với việc kiểm toán bản quyền khách hàng rất chặt chẽ.
* **PostgreSQL: Miễn phí và tự do**
  + **License:** Giấy phép PostgreSQL - một giấy phép mã nguồn mở tự do (tương tự BSD/MIT).
  + **Quyền lợi:**
    - Miễn phí sử dụng cho mọi mục đích (cá nhân, thương mại).
    - Miễn phí download, sửa đổi mã nguồn và phân phối lại.
    - **Không có chi phí license.**
  + **Tổng chi phí sở hữu (TCO):** Chi phí chủ yếu đến từ phần cứng, chi phí nhân sự (DBA) và chi phí hỗ trợ từ các công ty bên thứ ba (nếu cần) như EDB, Percona, Crunchy Data.

#### **Bảng so sánh chi phí (Ví dụ)**

| Hạng mục | Oracle Enterprise Edition | PostgreSQL |
| --- | --- | --- |
| **Chi phí License** | **Rất cao** (Hàng chục nghìn USD/core) | **$0** |
| **Tính năng Partitioning** | **Trả phí thêm** (Partitioning Option) | **Miễn phí**, có sẵn |
| **Tính năng HA (RAC/Replication)** | **RAC trả phí rất cao** | **Miễn phí** (Streaming Replication) |
| **Công cụ giám sát (AWR)** | **Trả phí thêm** (Diagnostics Pack) | **Miễn phí** (Dùng pg\_stat\_statements + tool khác) |
| **Mã nguồn** | Đóng | Mở |
| **Hỗ trợ** | Bắt buộc mua từ Oracle | Tùy chọn (Cộng đồng hoặc công ty thứ ba) |
| **Tổng chi phí sở hữu** | **Rất cao** | **Thấp** |