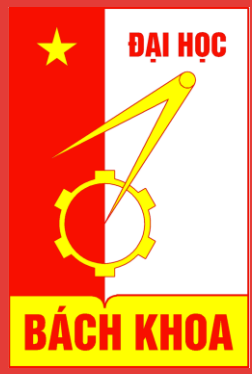


Quản trị cơ sở dữ liệu và Tối ưu hiệu năng

Tối ưu tương tranh (concurrency tuning)

Danh mục

1. Tối ưu tương tranh
   - Tối ưu khóa (Lock tuning)

Mục tiêu tối ưu tương tranh

- Mục tiêu về hiệu năng (Performance goals):
  - Giảm tắc nghẽn (reduce blocking) (Một giao dịch đợi các giao dich khác giải phóng khóa)
  - Tránh các bế tắc (deadlocks) và hủy giao dịch (rollbacks)
- Mục tiêu về tính chính xác (Correctness goals):
  - Tuần tự: Mỗi giao dịch thực hiện tách biệt
  - Lưu ý: Tính chính xác của chuỗi các thực hiện giao dịch phải phải được đảm bảo bởi lập trình viên.

Đánh đổi giữa tính chính xác các và hiệu năng !

Các ý tưởng trong giao dịch

- Lấy ít khóa
- Ưu tiên khóa chia sẻ hơn khóa đặc biệt
- Lấy các khóa với granularity tốt (fine granularity):
  - Các granularity: bảng, trang , hàng
  - Giảm phạm vi xung đột (conflict)
- Giữ các khóa trong thời gian ngắn:
  - Giảm thời gian chờ.

Tối ưu khóa
(Lock Tuning)

1. Loại trừ các khóa không cần thiết
2. Kiểm soát granularity của khóa (Control granularity of locking)
3. Loại bỏ các điểm nóng
4. Đảm bảo tính tách biệt và ảnh chụp tách bạch (snapshot isolation)
5. Chia tách các giao dịch dài.

1. Loại trừ các khóa không cần thiết

- Chi phí khóa (Lock overhead):
  - Memory: Lưu khóa điều khiển các tắc nghẽn
  - CPU: Xử lý các yêu cầu khóa
- Các khóa không cần thiết nếu:
  - Chỉ một giao dịch chạy tại một thời điểm, ví dụ: trong khi tải CSDL
  - Tất cả các giao dịch chỉ đọc, ví dụ: các truy vấn hỗ trợ quyết định trên dữ liệu lưu trữ.

2. Điều khiển Granularity của khóa (Control Granularity of Locking)

- Các khóa có thể được định nghĩa với các Granularity khác nhau:
  - Khóa mức hàng (cũng được gọi: khóa mức bản ghi)
  - Khóa mức trang
  - Khóa mức bảng
- Fine-grained locking (Mức hàng):
  - Hữu ích cho các giao dịch online ngắn
  - Mỗi giao dịch chỉ truy cập đến một vài bản ghi
- Coarse-grained locking (Mức bảng):
  - Tránh tắc nghẽn đối với các giao dịch dài
  - Tránh bế tắc
  - Giảm chi phí khóa

Leo thang khóa (Lok Escalation)

- Leo thang khóa (Lok Escalation): (SQL Server and DB2 UDB)
  - Tự động nâng cấp các khóa mức hàng thành các khóa của bảng nếu số lượng các khóa mức hàng đến một ngưỡng đã định trước.
  - Leo thang khóa có thể dẫn đến bế tắc.
- Oracle không hỗ trợ leo thang khóa.

Các thông số tối ưu granularity (Granularity Tuning Parameters)

- Cơ chế điều khiển rõ ràng (Explicit control) của granularity:
  - Bên trong giao dịch: Giao dịch yêu cầu một khóa mức bảng, khóa chia sẻ hoặc khóa đặc biệt (Oracle, DB2)
  - Phía bên kia giao dịch: lock granularity được định rõ cho mỗi bảng, tất cả các giao dịch truy cập vào bảng đó đều sử dụng chung một granularity (SQL Server)
- Thiết lập điểm leo thang (Escalation point):
  - Khóa được leo thang nếu số lượng các khóa mức hàng vượt qua một ngưỡng (điểm leo thang)
  - Điểm leo thang có thể được thiết lập bởi quản trị viên cơ sở dữ liệu
  - Quy tắc ngón tay cái (rule of thumb): Đủ cao để tránh leo thang cho các giao dịch trực tuyến ngắn.
- Kích thước bảng khóa (lock table):
  - Số lượng cực đại của tất cả các khóa có thể bị giới hạn
  - Nếu bảng khóa bị đầy, hệ thống buộc phải leo thang.

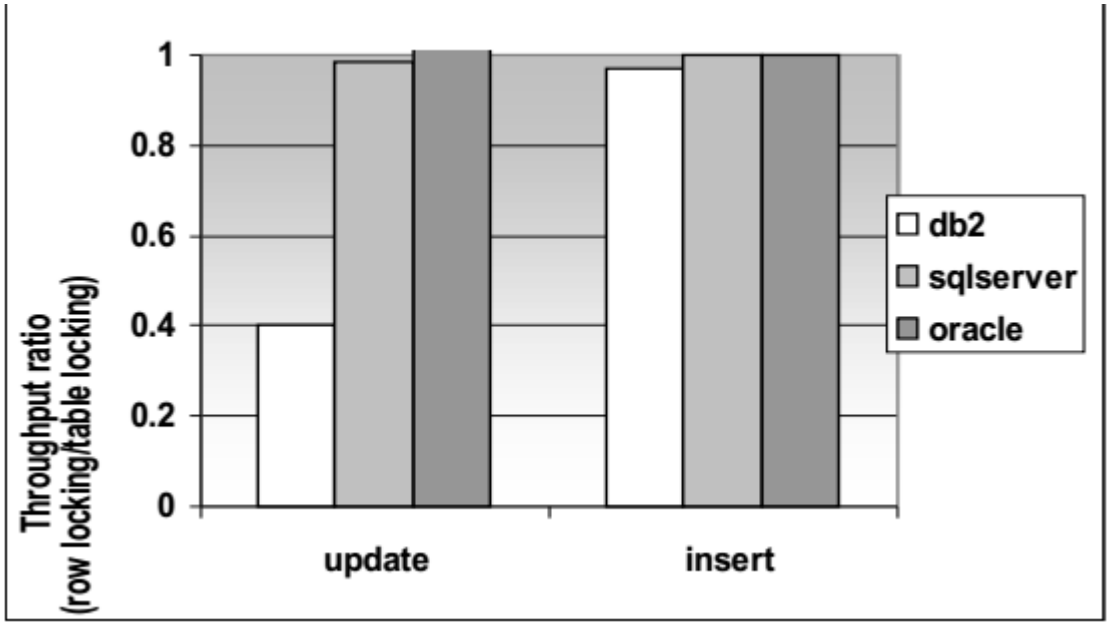
Chi phí của khóa bảng và khóa hàng (Overhead of Table vs. Row Locking)

- Cài đặt thực nghiệm:
- accounts(number,branchnum,balance)
- Chỉ số được phân nhóm (clustered index) trên number
- 100,000 hàng
- SQL Server 7, DB2 v7.1 và Oracle 8i trên Windows 2000
- Leo thang khóa bị tắt
- Truy vấn: (Không có các giao dịch đồng thời!)
  - 100,000 cập nhật (1 truy vấn)

    Ví dụ: Cập nhật bảng accounts với balance=balance*1.05
  - 100,000 chèn ((100,000 truy vấn)

    Ví dụ: chèn vào bảng accounts các giá trị (713,15,2296.12).

Chi phí của khóa bảng và khóa hàng (Overhead of Table vs. Row Locking)

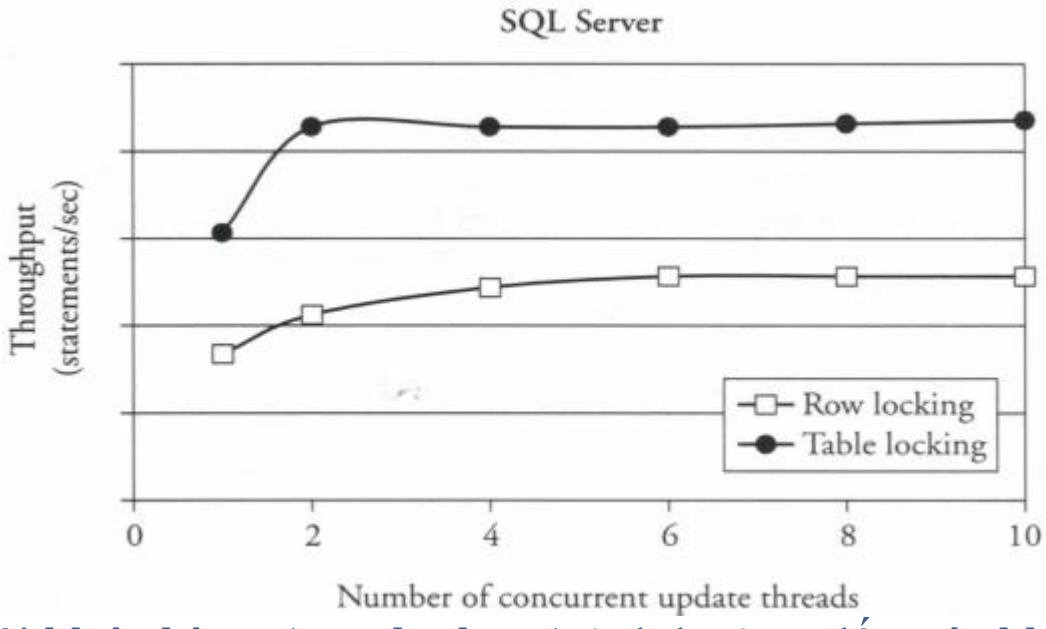


- Khóa hàng (Row locking) (100k hàng phải được khóa) đắt đỏ hơn khóa bảng (table locking) (1 bảng phải được khóa)
- SQL Server, Oracle: Phục hồi chi phí (ghi chép những thay đổi) ẩn đi sự khác biệt trong chi phí khóa
- DB2: Chi phí thấp do những ghi chép hợp lý các cập nhật, khác biệt rõ ràng về chi phí khóa

Thực nghiệm: Fine-Grained Locking

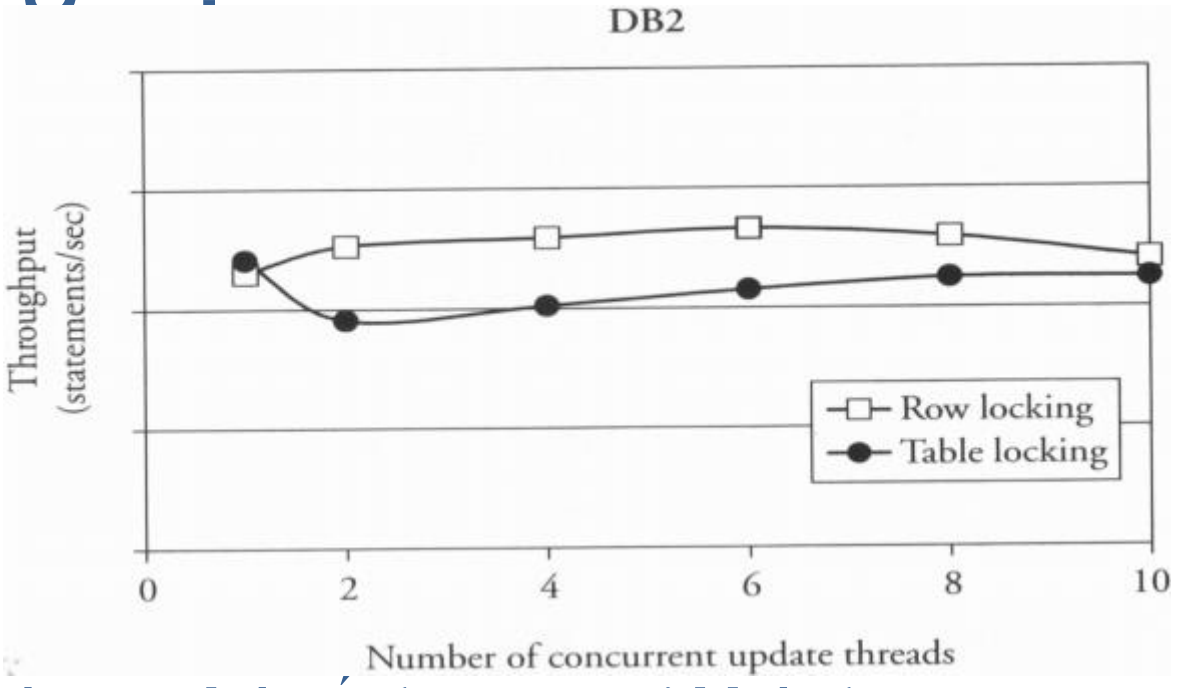
- Cài đặt thực nghiệm:
  - Bảng với các tài khoản ngân hàng
  - Chỉ số đực phân nhóm trên số tài khoản
  - Các giao dịch dài (Tổng của các số dư tài khoản)
  - Nhiều giao dịch ngắn (các chuyển khoản debit/credit)
  - Tham số: Số lượng các giao dịch đồng thời
  - SQL Server 7, DB2 v7.1 và Oracle 8i trên Windows 2000
  - Leo thang khóa bị tắt

Thực nghiệm: Fine-Grained Locking



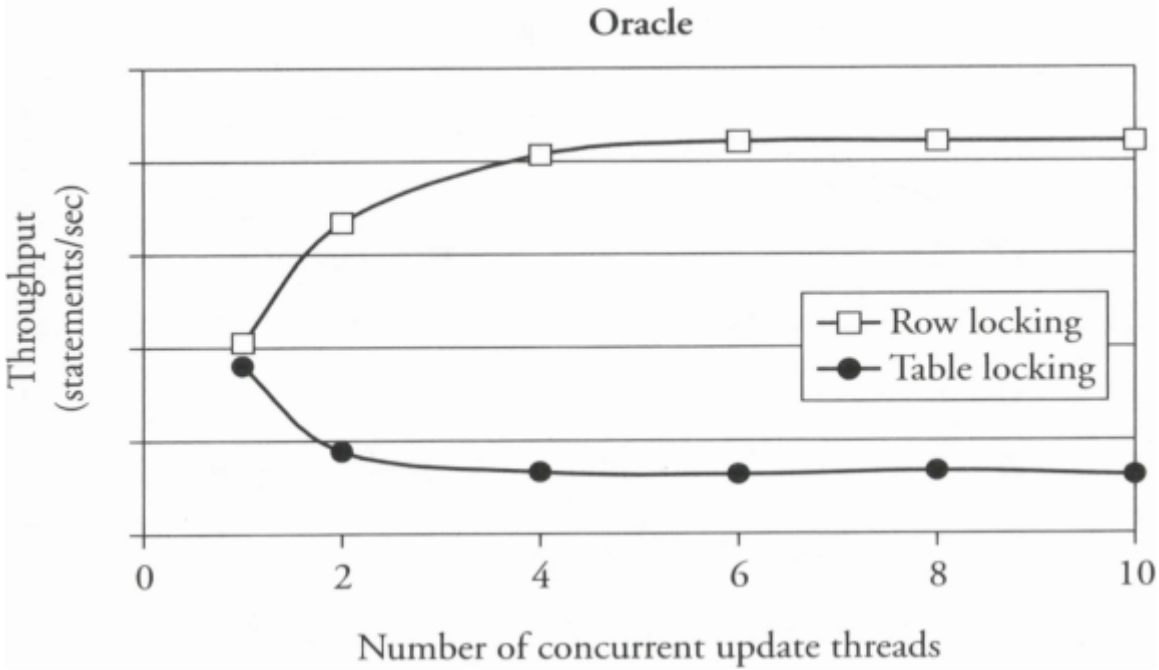
- Tính trình tự với khóa hàng (row locking) ảnh hưởng đến các khóa khoảng (key range locks)
- Các khóa khoảng được thực hiện trong chỉ số được phân nhóm.
- SQL Server: Chỉ số được phân nhóm có tính thưa thớt, do đó toàn bộ các trang bị khóa.
- Khóa mức hàng chỉ tăng một chút tính đồng thời
- Khóa bảng (Table-locking) ngăn ngừa phục hồi (rollback) cho truy vấn lấy tổng.

Thực nghiệm: Fine-Grained Locking



- Khóa hàng có một chút tốt hơn so với khóa bảng
- DB2 tự động chọn locking granularity nếu không có bất kì tinh chỉnh bằng tay
  - Tìm kiếm chỉ số (index scan) trong thực nghiệm này dẫn đến khóa mức hàng
  - Tìm kiếm bảng (table scan) có thể dẫn đến khóa mức bảng.

Thực nghiệm: Fine-Grained Locking



- Oracle sử dụng ảnh chụp tách bạch (Snapshot isolation):Truy vấn lấy tổng không bị xung đột với những giao dịch ngắn
- Khóa bảng: Các giao dịch ngắn phải đợi

3. Loại bỏ các điểm nóng (Circumvent Hot Spots)

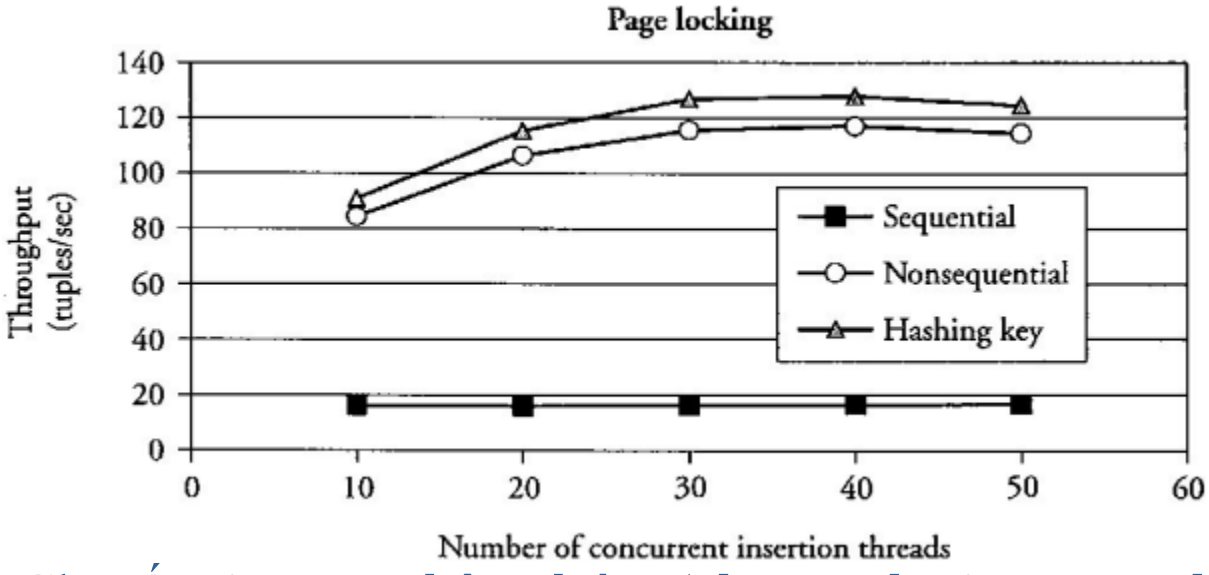
- Điểm nóng (Hot spot): Các mục dữ liệu có:
  - Được truy cập bởi nhiều giao dịch
  - Được cập nhật bởi ít nhất một vài giao dịch
- Loại bỏ các điểm nóng:
  - Truy cập điểm nóng càng muộn càng tốt trong giao dịch (giảm thời gian trờ cho các giao dịch khác vì các khóa bị giữ cho đến khi kết thúc các giao dịch)
  - Sử dụng phân vùng (partitioning), ví dụ: Nhiều danh sách tự do (multiple free lists)
  - Sử dụng phương tiện CSDL đặc biệt, ví dụ: Chốt trên bộ đếm(latch on counter)

Ví dụ phân vùng: các kiểu chèn được phân bổ

- Tranh chấp khi chèn (Insert contention): Trang bảng cuối cùng là nút thắt cổ chai:
  - Thêm dữ liệu vào heap file (ví dụ: các log file)
  - Chèn các bản ghi với các key trình tự (sequential keys) vào bảng với B^+ - tree.
- Giải pháp:
  - Sử dụng chỉ số băm đã phân cụm (clustered hash index)
  - Nếu chỉ có B^+ - tree có thể sử dụng, sử dụng thời gian chèn băm (hashed insertion time) như là key
  - Sử dụng khóa hàng thay vì khóa trang
  - Nếu việc đọc chỉ là tìm kiếm: Xác định nhiều điểm chèn (insertion points)

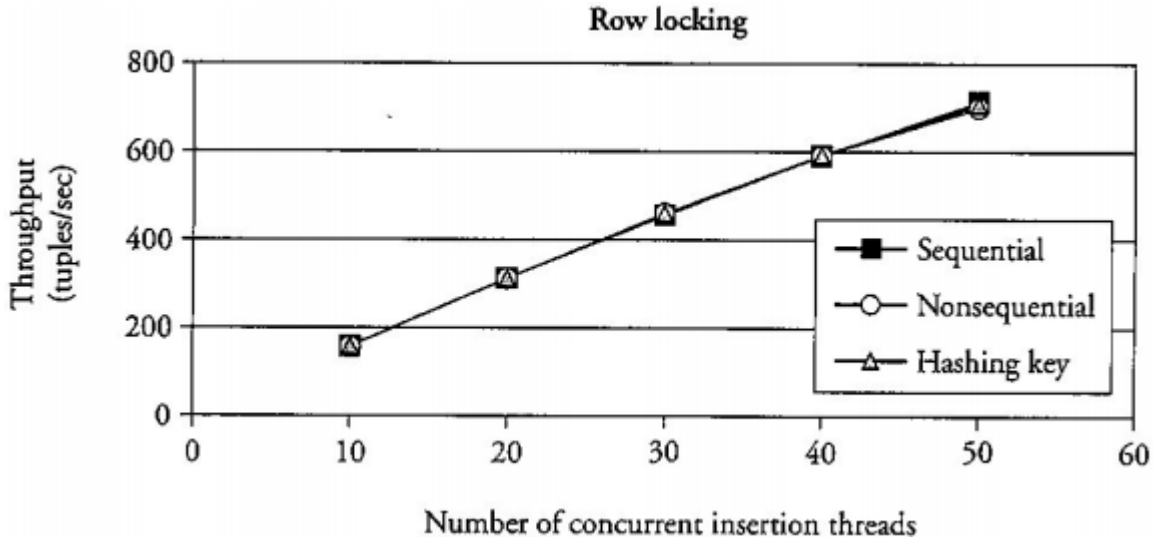
(Chỉ số tổng hợp (composite index) trên số nguyên ngẫu nhiêu (1..k) và thuộc tính key (key attribute))

Thực nghiệm: Đa điểm chèn và khóa trang



- Tính liên tục: Chỉ số B^+ - tree phân nhóm (clustered B^+ - tree index) và key theo thứ tự chèn.
- Tính không liên tục: B^+ - tree phân nhóm, và key độc lập với thứ tự chèn.
- Băm (Hashing): Chỉ số tổng hợp trên số nguyên ngẫu nhiên (1…k) và thuộc tính key.
- Khóa trang và các key liên tục: tranh chấp khi chèn!

Thực nghiệm: Đa điểm chèn và khóa hàng



- Không có tranh chấp khi chèn với khóa hàng

Ví dụ về phân vùng: Các lệnh DDL và danh mục

- Danh mục (Catalog): Thông tin về các bảng, ví dụ: tên, độ rộng cột
- Các lệnh ngôn ngữ định nghĩa dữ liệu (Data definition language (DDL) statements) phải đưa vào danh mục
- Danh mục có thể trở thành điểm nóng
- Phân vùng trong thời gian (Partition in time): Tránh được các lệnh DDL trong khi hoạt động của hệ thống hoạt động nặng

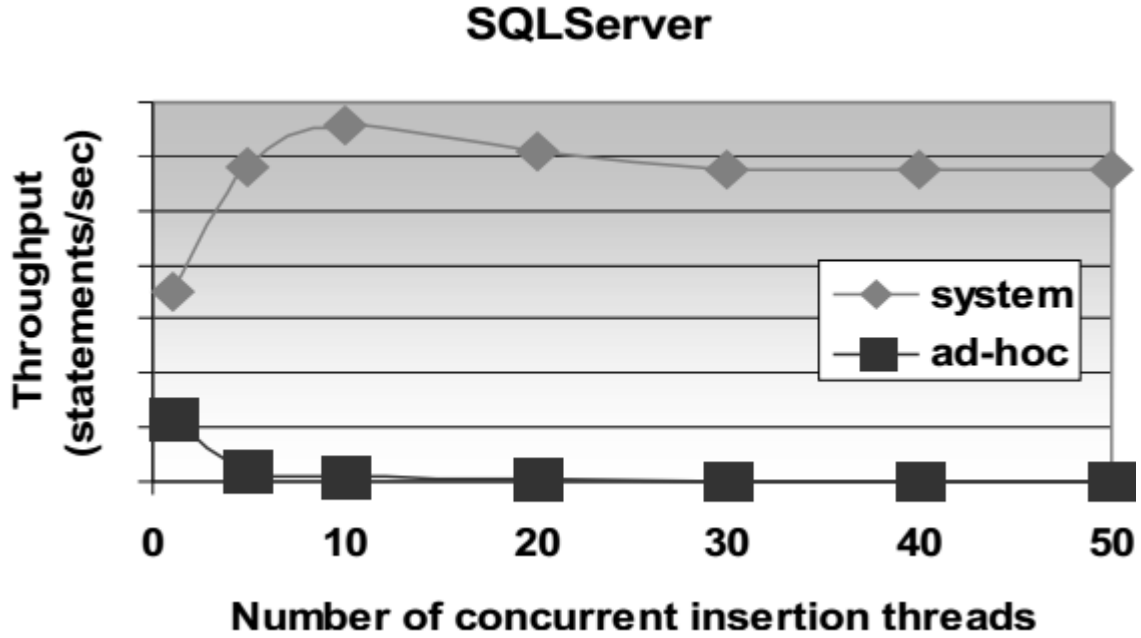
Ví dụ về phân vùng: Danh sách tự do (Partitioning Example: Free Lists)

- Tranh chấp khóa (Lock contention) trên danh sách tự do (free list):
  - Danh sách tự do: Danh sách các trang đệm CSDL không sử dụng
  - Một luồng cần một trang tự do (free page) khóa danh sách tự do
  - Trong khi các luồn khác không khóa có thể có được một trang tự do
- Giải pháp:
  - Tạo ra một vài danh sách tự do
  - Mỗi danh sách tự do chứa các con trỏ (pointers) trỏ đến một phần của các trang tự do
  - Một luồng cần một trang tự do chọn ngẫu nhiên một danh sách
  - Với n danh sách tự do việc tải mỗi danh sách được giảm bởi nhân tố 1/n

Phương tiện hệ thống: Chốt trên bộ đếm (System Facilities: Latch on Counter)

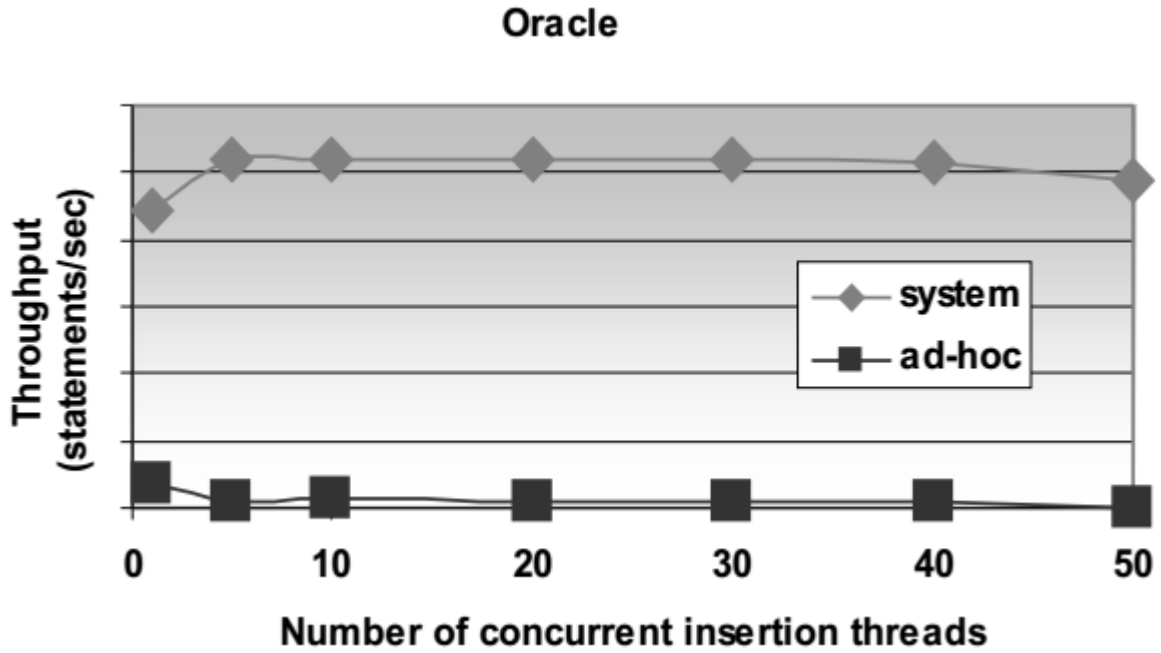
- Ví dụ: Chèn đồng thời với bộ định danh duy nhất
  - Bộ định danh được tạo ra bởi một bộ đếm (counter)
  - 2 giai đoạn khóa (2-phase locking): khóa trên bộ định danh được giữ đến khi giao dịch kết thúc.
  - Bộ định danh trở thành điểm nóng
- Các CSDL cho phép giữ một chốt trên bộ đệm (latch on the counter).
  - Chốt (latch): khóa đặc biệt chỉ được giữ khi truy cập
  - Loại bỏ được nút thắt cổ chai, nhưng có thể đưa vào những khoảng trống cho các giá trị bộ đệm
- Các khoảng trống với các chốt:
  - Giao dịch T1 tăng bộ đếm lên i
  - Giao dịch T2 tăng bộ đếm lên i + 1
  - Nếu hủy T1, sẽ không có mục dữ liệu có bộ định danh i

Thực nghiệm: Chốt và khóa trên bộ đếm (Latch vs. Lock on Counter)



- System (= chôt): Sử dụng phương tiện hệ thông đê sinh ra các giá trị bộ đếm (“Tính đồng nhất” (identity) trong SQL Server)
- Ad hoc (= khóa): Tăng một giá trị bộ đếm trong một bảng phụ (ancillary table)

Thực nghiệm: Chốt và khóa trên bộ đếm (Latch vs. Lock on Counter)



- System (=latch):Sử dụng phương tiện hệ thống để sinh ra các giá trị bộ đếm (“Tính liên tục” (sequence) trong Oracle)
- Ad hoc (= khóa): Tăng một giá trị bộ đếm trong một bảng phụ (ancillary table)