مختصر كتابي

Why SQL Is Not Enough



لماذا لا يكفي أن يعمل كل Query SQL بلا خطأ؟

تخيّل متجرًا إلكترونيًا بقيت فيه قطعة واحدة من سماعة معينة. ليلى وعمر فتحا صفحة المنتج من جهازين مختلفين، وضغطا زر الشراء في وقتين متقاربين جدًا. المتوقع أن ينجح طلب واحد فقط، وأن يُبلَّغ الآخر بنفاد المخزون.

لكن ما حدث أن البرنامج قرأ المخزون لطلب ليلى فوجد قطعة واحدة، وقبل أن يسجّل أي تغيير قرأ المخزون لطلب عمر فوجد القطعة نفسها. حسب كل طلب النتيجة عنده: واحد ناقص واحد يساوي صفرًا، ثم كتبها. النتيجة طلبان مقبولان والمخزون صفر، وليس سالبًا حتى؛ لذلك فإن منع الأرقام السالبة وحده لا يكفي هنا. السبب أن كل طلب اعتمد على قراءة قديمة وكتب النتيجة التي خرجت منها، ولا يلزم أن تحدث الخطوتان في اللحظة نفسها، بل يكفي أن تتداخل خطوات الطلبين بهذا الترتيب.

هذه الحالة، أي تداخل العمليات مع بعضها، تُسمّى **التزامن** (Concurrency). والمطلوب الآن أن تفهم أن ترتيب القراءة (Read) والكتابة (Write) جزء من صحة الحل، لا أن تحفظ حلًا جاهزًا.

وهناك مشكلة أخرى حتى مع عميل واحد: سجّل البرنامج الطلب وحفظه، ثم توقف فجأة قبل تحديث المخزون. هنا حُفظ جزء من العملية ولم يُحفظ الجزء الآخر. نريد أن تسير التغييرات معًا كوحدة واحدة: إما أن تتم كلها أو لا يتم شيء منها، وهذه الوحدة هي **المعاملة** (Transaction). لكن تجميع الخطوات في معاملة لا يحل وحده مشكلة ليلى وعمر، كما أن التراجع (Rollback) داخل الـDatabase لا يُعيد خصمًا تمّ عند شركة دفع خارجية.

وحين يكبر المتجر إلى مليون طلب، قد تفحص الـDatabase الطلبات واحدًا واحدًا لتجد طلبات ليلى القليلة. الفكرة وراء **الـIndex** تشبه دليل أسماء مرتبًا يدلّك على المكان مباشرة، لكنه يأخذ مساحة ويحتاج تحديثًا مع كل تغيير، لذلك نقيس قبل أن نقرر. وإن ضاعت البيانات، فإن **الـReplication** تساعد على استمرار الخدمة، لكنها قد تنقل الحذف الخاطئ نفسه؛ ولهذا نحتاج **نسخًا احتياطية** (Backups) نتأكد مسبقًا أننا نستطيع الاستعادة (Restore) منها.

- نجاح كل Query وحده لا يعني أن نتيجة النظام كله صحيحة.
- الاعتماد على قراءة قديمة ثم كتابة نتيجتها قد يبيع القطعة الأخيرة مرتين.
- المعاملة تجعل الخطوات تتم كلها أو لا يتم منها شيء، ولها حدود يجب فهمها.
- الـIndex يحل مشكلة أداء، ولا يمنع بيع آخر قطعة مرتين.
- الـReplica لا تُغني عن نسخ احتياطية مُجرَّبة الاستعادة.
- العمل العملي في الدورة على Postgres، بطريقة: نتوقع، ثم نجرّب على بيانات قليلة، ثم نلاحظ ونفسّر.

تحتاج قبل البدء إلى أساسيات SQL فقط: القراءة من Table، وإضافة Row وتعديله، وفكرة الربط (Join).

الخلاصة: هندسة الـDatabases تبدأ من المشكلة والضمان المطلوب: هل النتيجة صحيحة؟ كيف تتداخل العمليات؟ ماذا لو توقف العمل في منتصفه؟ كم يكلّف الوصول إلى البيانات؟ وهل نستطيع استعادتها إن

ضاعت؟ ثم نختار الـQuery والتصميم المناسبين.