مختصر كتابي

Primary and Candidate Keys



كيف نحدد صفًا واحدًا بعينه دون أي لبس؟

لدى خدمة العملاء طلبان، وصاحب كل منهما اسمه عمر محمد، وكتب الموظف: «عدّل عنوان عمر محمد». أيهما؟ الاسم مفيد للتعارف، لكنه لا يضمن تحديد شخص واحد.

Key هو عمود واحد أو عدة أعمدة معًا نميّز بها الصف عن أي صف آخر. و**Candidate Key** يمكن الاعتماد عليه: قيمه موجودة دائمًا، ولا يتكرر، ولا يحوي عمودًا زائدًا. فلو أضفنا الاسم إلى رقم العميل لما تغيّر شيء، أي أن الاسم عمود زائد.

إن كان لكل حساب بريد واحد لا يتكرر ولا يكون فارغًا، فالبريد مرشح ورقم العميل الداخلي مرشح آخر. نختار أحدهما **Primary Key**، ونحمي الآخر بمنع التكرار ومنع الفراغ. ونفضّل الرقم الداخلي لأن العميل قد يغيّر بريده، ونريد أن تبقى طلباته مرتبطة به. الرقم الذي نضيفه للتعريف هو **Surrogate Key**، والقيمة ذات المعنى في المجال، كرقم تسجيل رسمي، هي **Natural Key**. لا يوجد خيار أفضل دائمًا؛ نوازن بين الثبات والحجم والخصوصية وضمان التفرّد.

```
CREATE TEMP TABLE key_customers(
 id bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
 name text NOT NULL,
 email text UNIQUE NOT NULL
);
CREATE TEMP TABLE key_orders(
 id bigint PRIMARY KEY,
 public_code text UNIQUE NOT NULL,
 customer_id bigint NOT NULL REFERENCES key_customers
);
INSERT INTO key_customers(name,email) VALUES
 ('Omar Mohamed','omar1@example.test'),('Omar Mohamed','omar2@example.test');
INSERT INTO key_orders VALUES(100,'ORD-A100',1);
UPDATE key_customers SET email='omar.new@example.test' WHERE id=1;
```

شخصان بالاسم نفسه دون مشكلة، وبعد تغيير البريد يبقى الطلب 100 للعميل 1. بخاصية `IDENTITY` تولّد الـDatabase القيمة، لكن التوليد شيء والتفرّد شيء آخر؛ فالـPrimary Key هو ما يمنع التكرار والفراغ. وقد تكون في الأرقام المولّدة فجوات، فلا تحسب عدد الطلبات من آخر رقم، ولا تفترض أن الرقم الناقص يعني حذف طلب.

وليس كل جدول بحاجة إلى رقم جديد. في بنود الطلب يظهر المنتج مرة واحدة في الطلب، فنحدد البند برقم الطلب مع رقم المنتج، وهذا **Composite Key**. القيد ينظر إلى العمودين معًا: تكرار الزوج نفسه مرفوض حتى لو اختلف عدد القطع، أما منتج آخر في الطلب نفسه فمقبول.

```
CREATE TEMP TABLE key_items(
 order_id bigint REFERENCES key_orders,
 product_id bigint,
 quantity integer,
 PRIMARY KEY(order_id,product_id)
);
INSERT INTO key_items VALUES(100,101,2),(100,102,1);
INSERT INTO key_items VALUES(100,101,1); -- تكرار الزوج
```

- إن سمح المتجر بالمنتج مرتين في الطلب بخصمين مختلفين، نعطي كل بند رقمًا خاصًا؛ فقواعد المتجر هي التي تحدد الـKey.
- الـPrimary Key لا يعد بترتيب الصفوف عند القراءة؛ اكتب `ORDER BY` إن احتجت ترتيبًا.
- في Postgres لا يعني وجود الـKey أن الصفوف محفوظة على القرص بترتيبه.
- صعوبة تخمين رقم الطلب ليست بديلًا عن فحص الصلاحية: الهوية تحدد أي صف، والصلاحية تحدد من يراه.
- الـKey الكبير أو كثير التغيّر يؤثر على كل الجداول المرتبطة به.
- كود الفاتورة الظاهر للعميل إن كان يحدد طلبًا واحدًا فيجب منع تكراره وفراغه بنفسك، فالـKey الداخلي لا يمنع ذلك.

الخلاصة: اسأل دائمًا: إن غيّرنا هذه القيمة، هل ما زلنا نتحدث عن الكيان نفسه أم عن كيان جديد؟ الإجابة تحدد الـKey المناسب، وتفرّق بين اسم الصف وهويته والكود الظاهر للعميل وترتيب العرض.