En regardant ces deux migrations PostgreSQL, voici leur rôle précis :

Migration 1 : `add-computed-billing-status.cjs`

But : Automatiser le calcul du statut de facturation des employés selon des règles anti-fraude

Ce qu'elle fait :

- Crée les types ENUM nécessaires (`contractual_status_enum` , `leave_type_enum` , `billing_status_computed_enum`)
- **Remplace une colonne calculée par une VUE** : `xa_employee_license_with_billing_status`
- La vue applique automatiquement 5 règles métier :
  1. Pointage récent (< 7 jours) = BILLABLE
  2. Pas de pointage + en congé déclaré = NON_BILLABLE
  3. Pas de pointage + pas en congé + licence active = GRACE_PERIOD
  4. Licence terminée = TERMINATED
  5. Autres cas = NON_BILLABLE
- Ajoute des contraintes anti-fraude (pas de congé avec pointage récent, validation des données de congé)

Migration 2 : `add-generated-columns-global-license.cjs`

But : Automatiser le décompte des sièges achetés par licence globale

Ce qu'elle fait :

- Crée les types ENUM pour les licences (`license_type_enum` , `license_status_enum`)
- **Remplace une colonne calculée par une VUE** : `xa_global_license_with_seat_count`
- La vue calcule automatiquement `total_seats_purchased` en comptant les employés liés
- Ajoute des contraintes de validation (dates cohérentes, cycles de facturation valides)

Pourquoi des VUES au lieu de colonnes générées ?

Les deux migrations ont **abandonné l'approche des colonnes générées** (commentée en bas) au profit de **vues** pour éviter :

- Les références circulaires entre tables
- Les problèmes d'immutabilité des colonnes générées
- Les difficultés de maintenance

Résultat : PostgreSQL gère automatiquement les calculs complexes via les vues, sans stockage redondant, tout en appliquant les règles métier directement en base.