Dans une base de données SQLite, il arrive parfois que l’on soit amené à exécuter la commande PRAGMA integrity_check. Cette opération est un outil de vérification interne qui permet de s’assurer que la base de données est cohérente et qu’aucune corruption n’est présente. Même si SQLite est réputé pour sa robustesse et sa fiabilité, plusieurs situations peuvent entraîner des problèmes : Arrêts imprévus ou plantages : Si le système ou l’application qui utilise la base de données s’arrête brutalement, certaines écritures peuvent rester incomplètes et créer des incohérences. Problèmes matériels : Une défaillance du disque, des secteurs corrompus ou des problèmes de mémoire peuvent altérer les données stockées. Mises à jour ou migrations : Lorsqu’on migre ou modifie la structure d’une base existante, des erreurs peuvent survenir si les scripts ou outils de migration ne sont pas parfaitement fiables. Multiples accès concurrents : Bien que SQLite gère le verrouillage, des accès simultanés mal coordonnés peuvent provoquer des anomalies.

La commande PRAGMA integrity_check va parcourir toutes les tables et index de la base, vérifier les relations internes, les contraintes et s’assurer que les pages du fichier de base de données sont cohérentes. Elle renvoie « ok » si tout est correct, ou signale des erreurs précises en cas de corruption. Dans de nombreux cas, les erreurs détectées par PRAGMA integrity_check sont liées à des index corrompus plutôt qu’aux données elles-mêmes. SQLite fournit alors une solution simple : la commande REINDEX. En réindexant la base de données (ou des index spécifiques), on reconstruit les structures des index à partir des données réelles, ce qui résout souvent les incohérences détectées sans perdre de données. Ainsi, le processus classique de maintenance devient : Exécuter PRAGMA integrity_check pour identifier les éventuelles anomalies. Si des problèmes d’index sont détectés, exécuter REINDEX pour reconstruire les index corrompus. Refaire un PRAGMA integrity_check pour confirmer que la base est désormais cohérente. Cette approche permet de maintenir la fiabilité d’une base SQLite, même lorsqu’elle a été soumise à des conditions d’utilisation difficiles, tout en limitant le risque de perte de données.