

G3: Évolution des systèmes : analyse SMOKE DB :

Bardieux Arthur, Tang David, Vandeloise Mikel

- **Objectif** : Analyse détaillée des interactions et de l'évolution de la base de données SMOKE DB.
- **Méthodologie** : Inspection du code source Java et analyse des requêtes SQL.
- **Focus** :
 - Analyse des schémas physique, logique et conceptuel.
 - Étude de la fréquence, diversité et complexité des interactions.
- **Thèmes clés** :
 - Optimisation et performance.
 - Maintenance et évolutivité.
 - Sécurité et confidentialité des données.
- **But possible** : Améliorer l'efficacité, la robustesse et la sécurité de SMOKE DB.

Étape 1: Analyse des accès

Méthodologie et schéma physique

Méthodologie:

- Inspection détaillée du code source Java pour identifier les points d'accès à la base de données.
- Analyse approfondie des requêtes SQL intégrées pour comprendre les interactions avec la base de données.
- Utilisation de SQLInspector pour extraire et analyser les modèles d'accès aux données, en mettant l'accent sur les requêtes et leur fréquence.

Schéma physique:

- Identification de 12 tables distinctes et 98 attributs au total dans la base de données Smoke. Pas (ou peu) de présence d'index.

Étape 1: Analyse des accès

Méthodologie et schéma physique

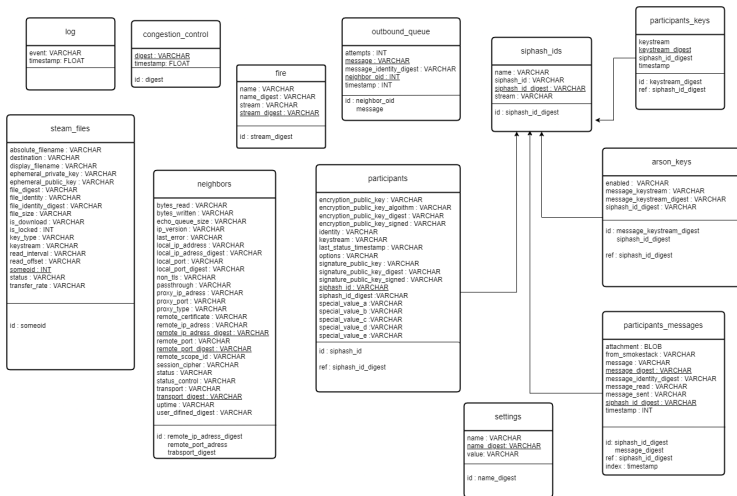


Figure: Diagramme physique

Étape 1: Analyse des requêtes SQL et optimisation

Schéma logique, conceptuel et sécurité

Schéma logique:

- Ajout de l'attribut "oid" dans les tables fondé sur l'analyse SQL.
- Schéma logique comparable au schéma physique avec enrichissements.
- Définition des clés primaires et étrangères dans les requêtes DDL, fournissant la structure relationnelle entre les tables.
- Analyse spécifique des tables 'log', 'outbound_queue', 'arsons_keys', 'participants_messages', et 'neighbors' pour leur rôle unique et structure complexe, notamment les identifiants composites et les relations inter-tables.

Schéma conceptuel:

- Transformation des clés étrangères "siphash_id_digest" en relations simples.
- Modification des clés dans les tables comme "participants_messages".

Optimisation et sécurité:

- Identification des requêtes complexes pour optimisation.
- Importance de la gestion sécurisée des clés et identifiants.
- Recommandations pour renforcer les mesures de sécurité.

Étape 1: Analyse des requêtes SQL et optimisation

Schéma logique

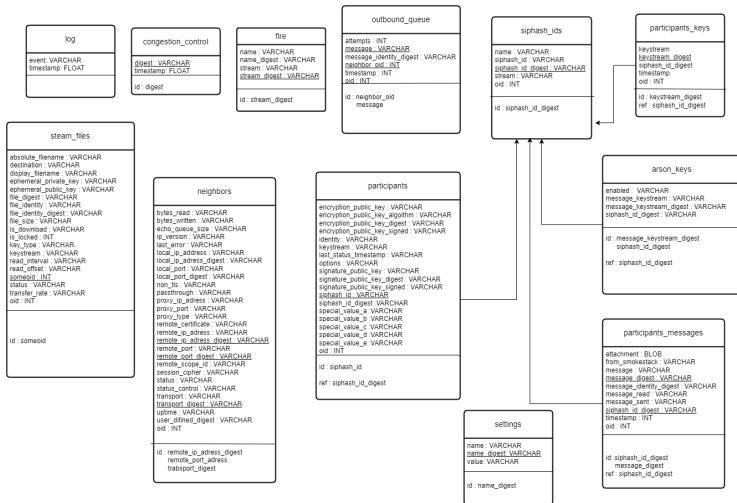


Figure: Diagramme logique

Étape 1: Analyse des requêtes SQL et optimisation

Schéma conceptuel

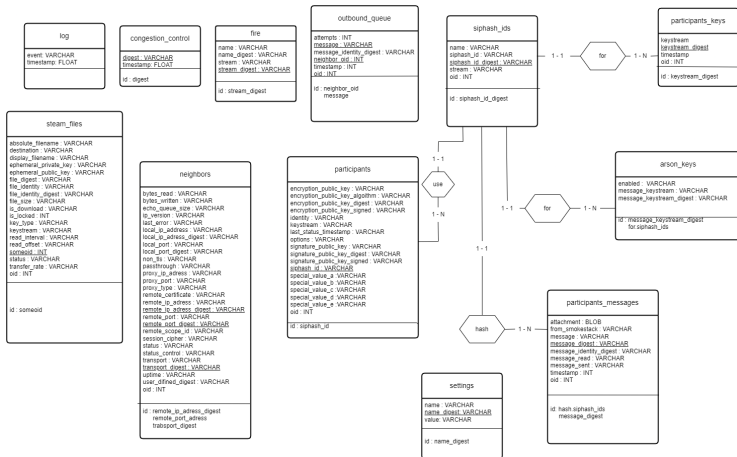


Figure: Schéma conceptuel

Étape 2: Méthodologie et analyse des requêtes

Approche détaillée et exemples spécifiques

Méthodologie:

- Analyse des requêtes SQL intégrées dans le code source Java.
- Focus sur l'identification des patterns d'accès et la complexité.

Répartition des Requêtes (Selects vs Autres)

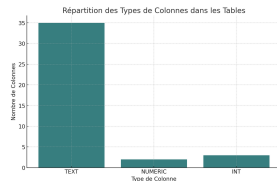
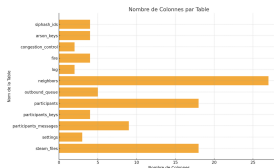
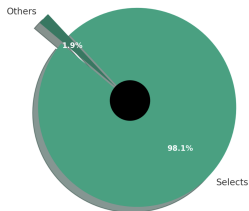


Figure: Statistiques des requêtes, tables et colonnes

Étape 2: Dérivation du LSS et recommandations

Implications stratégiques et améliorations proposées

Analyse des requêtes:

- Fréquence et utilisation des tables `neighbors`, `participants`, `participants_keys`.
- Complexité mise en évidence par des requêtes sur `neighbors` avec sous-requêtes imbriquées.
- Jointures complexes entre `participants` et `participants_keys` pour la gestion des identités.

Sous-schéma logique (LSS):

- Hiérarchisation fondée sur l'analyse des requêtes.
- Rôle clé des tables `participants`, `neighbors`, `siphash_ids`, `steam_files`.

Étape 2: Dérivation du LSS et recommandations

Implications stratégiques et améliorations proposées

Implications et recommandations:

- **Optimisation des requêtes** : Réécriture des sous-requêtes, utilisation de vues matérialisées.
- **Indexation pour performance** : Indexation des colonnes clés pour des requêtes plus rapides.
- **Sécurité des données** : Renforcement des protocoles de cryptage et sécurisation des requêtes sensibles.

Conclusion:

- Un aperçu détaillé de l'utilisation des tables et des requêtes critiques.
- Base solide pour l'optimisation future et la maintenance systématique.

Étape 3: Analyse d'impact des scénarios d'évolution

Méthodologie et détails des scénarios

Méthodologie:

- Analyse détaillée des impacts des scénarios d'évolution du PSS.
- Focus sur les modifications nécessaires dans le code source.
- Évaluation de la cohérence globale du système.

Scénarios d'évolution:

- ➊ Renommage de colonnes dans 'neighbors'.
- ➋ Ajout de colonnes de sécurité dans 'participants'.
- ➌ Suppression de colonnes dans 'siphhash_ids'.
- ➍ Division de la table 'steam_files'.
- ➎ Fusion de 'log' et 'events'.

Impacts des scénarios:

- Modifications dans les classes et méthodes spécifiques.
- Ajustements nécessaires pour maintenir la fonctionnalité.
- Évaluation des répercussions sur la maintenance et l'efficacité.

Étape 3: Implications et conclusion

Implications des scénarios et conclusions clés

Scénarios d'évolution (suite):

- ⑥ Modification des types de données dans 'settings'.
- ⑦ Introduction d'une table d'association 'users_groups'.
- ⑧ Changement de stratégie d'indexation sur 'participants_keys'.
- ⑨ Restructuration des clés primaires et étrangères.
- ⑩ Optimisation des requêtes pour 'neighbors'.

Implications des scénarios:

- Besoins variés en termes de mise à jour du code.
- Impacts potentiels sur la performance et l'intégrité des données.
- Nécessité de renforcer la sécurité pour les informations sensibles.

Conclusion:

- Compréhension claire des impacts des changements de schéma.
- Identification des ajustements pour la cohérence et l'efficacité.
- Fondation pour les décisions futures de développement et maintenance.

Étape 4: Évaluation de la qualité du schéma de la base de données

Points forts et axes d'amélioration

Points forts:

- **Cohérence et normalisation** : Structure de données bien organisée, utilisation efficace des clés primaires et étrangères.
- **Clarté des nomenclatures** : Noms des tables et colonnes explicitement définis, facilitant la compréhension du schéma.

Axes d'amélioration:

- **Optimisation des index** : Analyse des requêtes pour améliorer l'indexation et les performances.

Étape 4: Évaluation du code et recommandations

Points forts, axes d'amélioration et conclusion

Évaluation du code:

- **Structuration et organisation** : Code bien structuré et organisé pour une maintenance aisée.
- **Gestion des exceptions** : Gestion des erreurs, assurant la robustesse de l'application.

Axes d'amélioration et recommandations:

- **Optimisation des requêtes SQL** : Révision des requêtes complexes pour améliorer la performance.
- **Sécurité des requêtes** : Renforcement de la prévention contre les injections SQL.
- **Recommandations globales** : Simplification des requêtes, révision de l'indexation et renforcement de la sécurité.

Bonus: Complexité des requêtes et risques

Évaluation de la complexité et des risques de sécurité

Complexité croissante des requêtes:

- Évolution de requêtes simples vers des structures plus complexes avec sous-requêtes et jointures.
- Risques liés à la performance et à la sécurité en raison de la complexité accrue.

Sécurité et confidentialité:

- Utilisation de techniques cryptographiques comme *Base64.encodeToString* et *cryptography.hmac*.
- Risques liés à un mauvais usage de la cryptographie ou une gestion insuffisante des clés.

Optimisation des performances:

- Nécessité d'une optimisation continue pour éviter les problèmes de performance.
- Importance de l'attention constante aux requêtes fréquentes ou mal structurées.

Bonus: Adaptabilité, cohérence et conclusion

Défis de l'adaptabilité et de la maintenance

Adaptabilité et évolutivité:

- Adaptation aux nouveaux besoins, mais risques liés à un manque de planification durable.
- Équilibre entre évolutivité et stabilité du système.

Cohérence et maintenance:

- Importance de maintenir des normes de développement cohérentes.
- Nécessité d'une stratégie de maintenance bien définie pour assurer la stabilité à long terme.

Conclusion de l'étape bonus:

- Analyse révélant des améliorations, mais également des préoccupations en matière de performance, sécurité, adaptabilité et maintenance.
- Nécessité d'une approche plus mesurée et d'une analyse approfondie pour évaluer la qualité globale de l'application.