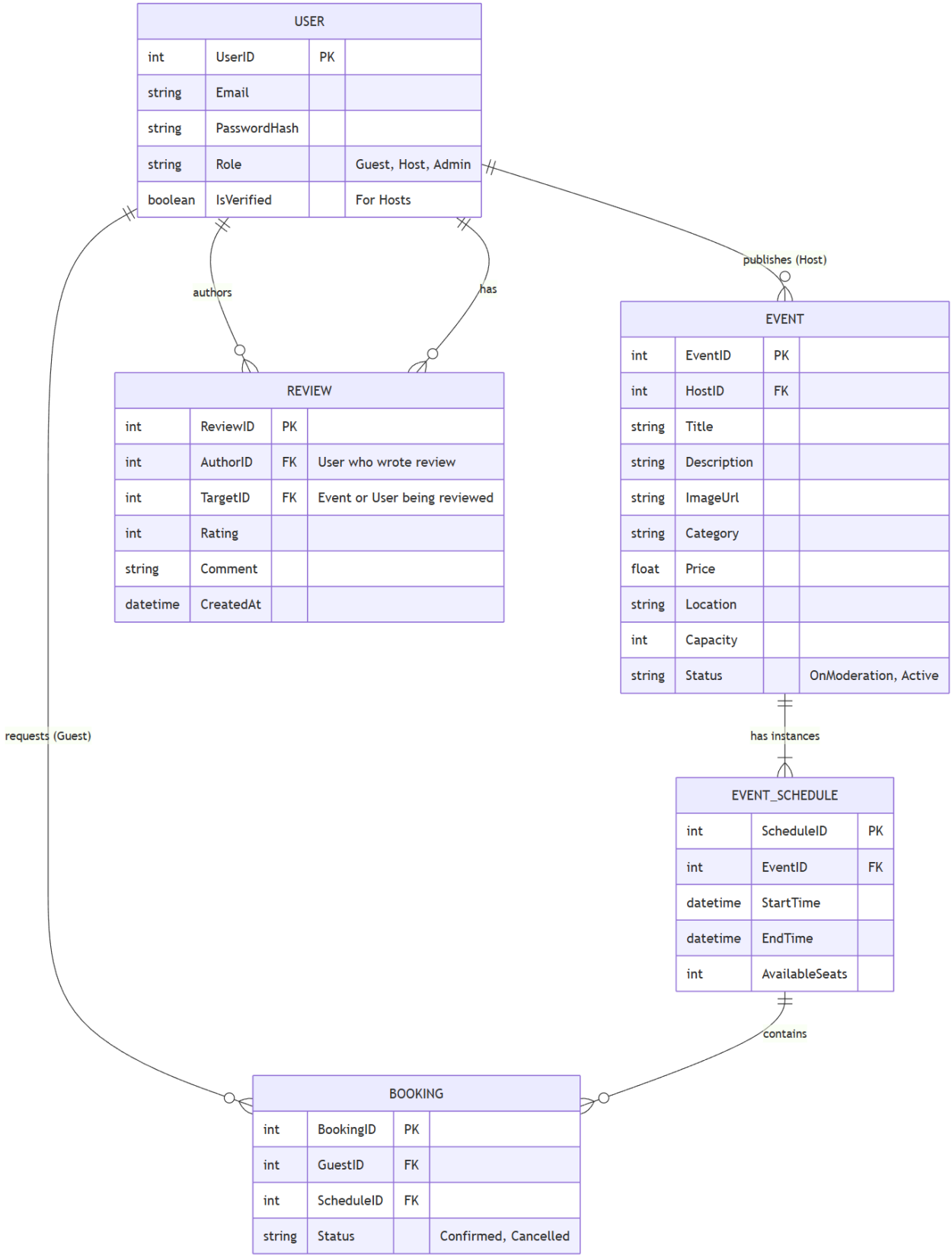
ER-діаграма (Crow’s Foot Notation)



Data Retention Policy			
Тип даних	**Період зберігання**	**Мета**	**Місце зберігання**
Системні логи	30 днів абодо досягнення ліміту 100 МБ	Налагодження, моніторинг безпеки, журнал аудиту	Централізована система логування
Профіль користувача	До видалення аккаунту користувачем	Ідентифікація користувача, доступ до сервісу, автентифікація	PostgreSQL (User)
Подія	Через 3 роки після завершення заходу	Вирішення конфліктів	PostgreSQL (Event)
Розклад події	Через 3 роки після завершення запланованої події	Вирішення конфліктів	PostgreSQL (Event_Schedule)
Бронювання	3 роки після завершення забронюваного заходу	Вирішення конфліктів	PostgreSQL (Booking)
Відкуг	Поки аккаунти користувача та хоста, які стосуються відгука, не є видаленими	Система довіри, історія репутації	PostgreSQL (Review)

Consensus, Batching, Streaming, Consistency

Для забезпечення consistency між розподіленими компонентами застосунку було обрано алгоритм **RAFT.** Даний алгоритм забезпечує надійне лідерство в кластері, реплікацію журналу операцій та підтвердження змін більшістю вузлів, що гарантує відсутність конфліктів при одночасних запитах на зміну критичних даних. Використання **RAFT** дозволяє забезпечити коректну роботу сервісів, що відповідають за бронювання, управління подіями та доступністю ресурсів, а також підвищує відмовостійкість системи у разі збоїв окремих вузлів. Наявність готових рішень робить цей підхід доцільним для нашого проекту.

Обробка даних реалізується у режимі **streaming**, що дозволяє виконувати обробку подій у реальному часі. Такий підхід чудово підходить для оновлення доступності подій та надсилання сповіщень, вже не кажучи про збір аналітики та моніторинг.

Критичні операції можна обмежити транзакціями для забезпечення сильної **консистентності**, що дозволить нашому застосунку бути відмовостійким, коли це має значення, тоді як інші операції за рахунок стрімінгу будуть досить швидкими, що позитивно вплине на швидкодію всього застосунку.

CID діаграма: