

# **Бази даних**

## **Лекція 13**

---

# Тематика лекції

- Денормалізація
- Організаційні питання

# Нормалізація

---

**Нормалізація - процес декомпозиції таблиць для уникнення надлишковості та небажаних аномалій даних.**

**Основні цілі нормалізації:**

- **простота модифікації даних**
- **консистентність даних**
- **простота підтримки бази даних**
- **мінімізація дублювань даних**

---

# Переваги нормалізації

- Простіше оновлення даних
- Простіша підтримка
- Простіші запити
- Оптимізація об'єму даних
- Консистентність даних

---

# Недоліки нормалізації

- Складніші запити - більше join-ів
- Більше таблиць - складніша логіка
- Нижча швидкодія читання

# Денормалізація

---

Денормалізація - процес **наміреного** внесення надлишковості до нормалізованих таблиць.

Основні цілі денормалізації:

- збільшення швидкодії запитів читання даних
- спрощення складних запитів читання даних

# Додавання додаткових колонок

---

Один з основних підходів денормалізації.

Полягає в додаванні додаткових колонок до таблиць, з яких часто відбувається читання з великою кількістю JOIN операцій.

Найкраще підходить для отримання даних, що часто читаються, проте дуже рідко змінюються.

# Додавання додаткових колонок

---

Order	
PK	order_id
	customer_id
	customer_name
	customer_city
	customer_zip
	customer_country
	country_tax
	total_amount

```
SELECT
c.name, c.country, c.city, o.total_amount
FROM order o
INNER JOIN customer c
USING (customer_id)
WHERE ...;
```

```
SELECT
customer_name, customer_country, customer_city, total_amount
FROM order
WHERE ...;
```

# **Додавання похідних колонок**

---

**Підхід полягає в додаванні додаткових похідних колонок до таблиць, з яких часто відбувається читання агрегаціями (GROUP BY).**

**Полягає в виборі між складністю читання з GROUP BY та складністю оновлень похідної колонки при будь-яких пов'язаних записах та оновленнях.**

# Додавання ПОХІДНИХ КОЛОНОК

---

Order	
PK	order_id
	customer_id
	customer_name
	customer_city
	customer_zip
	customer_country
	country_tax
	total_amount

```
SELECT
o.order_id, SUM(oi.price) as total_amount
FROM order o
LEFT JOIN order_items oi
USING (order_id)
WHERE ...
GROUP BY o.order_id;
```

```
SELECT
order_id, total_amount
FROM order
WHERE ...;
```

# Materialized views

---

**Materialized views дозволяють зберігати попередньо обчислені результати складних статистичних агрегацій, для яких не потрібні найновіші дані.**

**Підхід полягає в створенні materialized views, які зберігають складні статистичні вибірки у “закешованому” стані та допомагають уникнути довгих запитів для отримання статистики.**

# Materialized views

---

**CREATE MATERIALIZED VIEW** order\_summary **AS**

**SELECT**

o.order\_id,  
c.customer\_name,  
COUNT(oi.item\_id) as item\_count,  
SUM(oi.price) as total\_amount

**FROM** orders o

**JOIN** customers c **USING** (customer\_id) **J**

**JOIN** order\_items oi **ON** **USING**(order\_id)

**GROUP BY** o.order\_id, c.customer\_name;

**REFRESH MATERIALIZED VIEW CONCURRENTLY** order\_summary;

# Array та JSON колонки

---

**Для даних, що завжди зберігаються та завжди читаються “одним шматком” можлива денормалізація шляхом створення колонок типу JSON та масивів у реляційних базах даних.**

**Даний підхід дозволяє використовувати реляційні бази даних у форматі схожому на нереляційні документо-орієнтовані бази даних.**

# Денормалізація vs ІНДЕКСИ

---

Індекси для:

- Пришвидшення WHERE
- Пришвидшення JOIN, де це можливо
- Забезпечення унікальності записів (UNIQUE)
- Пришвидшення сортування

# Денормалізація vs ІНДЕКСИ

---

Денормалізація для:

- Усування JOIN-и
- Попереднього обчислення агрегацій
- Спрощення запитів

# Денормалізація vs індекси - рішення

---

Якщо запит повільний:

1. EXPLAIN ANALYSE
2. Додати індекси якщо їх не вистачає
3. Якщо індекси не спрацювали - денормалізовувати
4. Додати індекси для денормалізованої таблиці якщо потрібно

# ПРАКТИЧНІ ПОРАДИ

---

- Варто завжди аналізувати проблемні запити перед тим, як робити будь-які рішення щодо оптимізації.
- Завжди варто приймати рішення щодо нормалізації та денормалізації обдуманно та “стратегічно”.
- Денормалізація даних призводить до того, що підтримка бази даних стає складнішою.

# ПРАКТИЧНІ ПОРАДИ

---

- Денормалізація даних призводить до того, що одні і ті ж самі дані доводиться зберігати багато разів.
- Денормалізація помітно ускладнює записи та спрощує читання.
- Денормалізовані дані вимагають підтримки та можуть призвести до неконсистентностей даних.

# **Організаційні питання**