Database Table Design, Normalization & Indexing

Using FoodieExpress 🍔

---

1. Database Table Design (Basic Rules)

Good table design means: - No duplicate data - Clear primary keys - Proper foreign keys - Small, meaningful columns

Core Tables

Table	Purpose
customers	Store customer info
customer_profiles	Extra details of customers
restaurants	Restaurant data
menu_items	Food items of restaurant
orders	Orders placed
order_items	Items inside orders

---

2. Normalization (Simple Language)

1NF (First Normal Form)

Rules: - No repeating groups - Each column holds atomic values

❌Wrong: | order_id | items | | 1 | Pizza,Burger |

✅Correct: | order_id | item_id | | 1 | 101 | | 1 | 102 |

2NF (Second Normal Form)

Rules: - Must be in 1NF - No partial dependency

❌Wrong: | order_id | item_id | item_name | item_name depends only on item_id

✅Split: - menu_items(item_id, item_name) - order_items(order_id, item_id)

3NF (Third Normal Form)

Rules: - Must be in 2NF - No transitive dependency

❌Wrong: | customer_id | city | state | (state depends on city)

✅Split: - cities(city_id, city_name, state) - customers(customer_id, city_id)

---

3. Indexes (Fast Search)

Index = Book index → Faster lookup

Where to Add Index

Column	Reason
customers.email	login search
orders.customer_id	fetch orders
menu_items.restaurant_id	menu listing
order_items.order_id	order details

---

4. SQL Index Example

```
CREATE INDEX idx_orders_customer ON orders(customer_id);
CREATE INDEX idx_menu_restaurant ON menu_items(restaurant_id);
```

---

5. JPA Entity Mapping with Index

```
@Entity
@Table(name = "orders", indexes = {
    @Index(name = "idx_customer", columnList = "customer_id")
})
public class Order {
    @Id
    @GeneratedValue
    private Long orderId;
```

```
    @ManyToOne
    @JoinColumn(name = "customer_id")
    private Customer customer;
}
```

---

6. Benefits

- Faster queries
- Clean data
- Easy reporting
- No redundancy

Use this in interviews or training.