DineFlow Database Management System

Project Name: DineFlow
Project Type: Fast Casual Restaurant Management System
Date: December 10, 2024
Prepared By: Mansi Uday Bhavsar

Executive Summary

DineFlow is a comprehensive database management system designed for fast-casual restaurant operations. The system manages the complete customer journey from order placement through payment, food preparation, and delivery. This document presents a detailed analysis of user requirements, functional specifications, business rules, and the foundational entity model that will drive the database design.

The system supports four primary stakeholder groups: Customers (including loyalty program members), Cashiers, Kitchen Staff, and Waiters, with managerial oversight capabilities. The design emphasizes operational efficiency, order accuracy, and customer satisfaction through integrated loyalty programs and real-time order tracking.

Table Of Content

1. Stakeholder Analysis

1.1 Primary Stakeholders

1.1.1 Customers

Description: End users who purchase food and beverages from the restaurant.

User Types:

- **Guest Customers:** First-time or non-registered customers without loyalty accounts
- **Loyalty Members:** Registered customers with phone numbers earning rewards
- **VIP Members:** High-value customers with 100+ stars receiving premium benefits

Primary Needs:

- Fast, accurate order placement
- Transparent pricing and payment processing
- Loyalty reward tracking and redemption
- Order customization capabilities
- Reasonable wait times (≤30 minutes)

1.1.2 Cashiers

Description: Front-line staff responsible for customer interaction, order entry, and payment processing.

Primary Responsibilities:

- Customer greeting and order taking
- Menu navigation and item selection
- Special instruction documentation
- Payment processing (cash and card)
- Receipt generation
- Refund request initiation
- Loyalty program enrollment

System Access Level: Standard user with order creation and payment processing privileges

1.1.3 Kitchen Staff (Chefs)

Description: Personnel responsible for food preparation and order fulfillment.

Primary Responsibilities:

- Order queue monitoring
- Food preparation following specifications
- Special instruction compliance
- Order status updates (Queued → In Process → Completed)
- Time management to meet 30-minute SLA
- Quality assurance

System Access Level: Kitchen display access with order status modification rights

1.1.4 Waiters/Runners

Description: Service staff responsible for order delivery and table management.

Primary Responsibilities:

- Table assignment monitoring (maximum 6 tables per waiter)
- Order delivery from kitchen to customer
- Table status management (occupied/available)
- Customer service and issue escalation
- Delivery confirmation

System Access Level: Standard user with table management and delivery tracking privileges

1.1.5 Managers

Description: Supervisory staff with oversight and administrative capabilities.

Primary Responsibilities:

- Refund approval/denial
- Menu item management (add/edit/availability)
- Performance reporting and analytics
- Issue resolution and escalation handling

- Business rule enforcement
- Daily reconciliation

System Access Level: Administrative access with override capabilities

1.2 Secondary Stakeholders

Restaurant Owners: Financial oversight and strategic decision-making
Accountants: Financial reporting and tax compliance
Health Inspectors: Compliance verification (future consideration)

2. Functional Requirements

2.1 Customer Requirements

ID	Priority	Requirement Description
FR-C-001	Critical	Customer can provide phone number to become/identify as loyalty member
FR-C-002	Critical	System displays customer's current star balance when phone number entered
FR-C-003	Critical	Customer receives 1 star per order (regardless of order value)
FR-C-004	Critical	Customer receives automatic discounts: 10★=5% off, 25★=10% off, 50★=15% off, 100★=20% off
FR-C-005	Critical	Stars are deducted when discount is applied
FR-C-006	High	Customer can view order history (last 20 orders) when phone number entered
FR-C-007	Critical	Customer receives printed receipt with order number, items, prices, payment method, timestamp, estimated wait time, table number (if dine-in)

FR-C-008	High	Customer can request order modification within 5 minutes of order creation AND only if order status = 'QUEUED'
FR-C-009	High	Customer can specify special instructions per menu item (max 200 characters)
FR-C-010	Medium	System tracks customer preferences for future reference
FR-C-011	Critical	Customer can choose dine-in or takeout at time of order

2.2 Cashier Requirements

ID	Priority	Requirement Description
FR-CA-001	Critical	Cashier must log in with unique Cashier ID at start of shift
FR-CA-002	Critical	Cashier can browse menu items by category (Appetizers, Mains, Desserts, Beverages)
FR-CA-003	Critical	Cashier can add items to order with quantity (1-99)
FR-CA-004	Critical	Cashier can remove items from order before payment
FR-CA-005	High	Cashier can add special instructions to individual items
FR-CA-006	Critical	System displays real-time order total including tax (10% tax rate)
FR-CA-007	Critical	Cashier can apply loyalty discount if customer eligible

FR-CA-008	Critical	Cashier can process payment via Cash, Credit Card, or Debit Card
FR-CA-009	Critical	Only ONE payment method per order (no split payments)
FR-CA-010	Critical	System generates order number sequentially per day (format: YYYYMMDD-####)
FR-CA-011	Critical	Cashier can assign table number for dine-in orders
FR-CA-012	High	Cashier can initiate refund request (requires manager approval)
FR-CA-013	Medium	Cashier can reprint receipt if customer requests
FR-CA-014	High	Cashier logs out at end of shift
FR-CA-015	Medium	System tracks all orders by Cashier ID for performance review
FR-CA-016	High	Cashier can search customer by phone number to retrieve loyalty info
FR-CA-017	Medium	Cashier can view current queue length and estimated wait time
FR-CA-018	High	Cashier confirms special instructions verbally with customer before payment
FR-CA-019	High	Split payment handled by creating separate orders with linkage

2.3 Kitchen Staff Requirements

ID	Priority	Requirement Description
FR-K-001	Critical	Kitchen display shows orders in FIFO (First In, First Out) queue
FR-K-002	Critical	Chef can view order details: order number, items, quantities, special instructions, order time, table number
FR-K-003	Critical	Chef can mark order status: QUEUED → IN_PROCESS → COMPLETED
FR-K-004	High	Chef can view estimated prep time per item
FR-K-005	Critical	System alerts chef when order exceeds 25 minutes (warning) and 30 minutes (critical)
FR-K-006	Critical	Chef can notify waiter when order is COMPLETED
FR-K-007	Medium	System displays total prep time estimate for entire order
FR-K-008	High	Kitchen display color-codes orders by priority: Green (<15min), Yellow (15-25min), Red (>25min)
FR-K-009	Low	Chef can add notes to order

2.4 Waiter Requirements

ID	Priority	Requirement Description
FR-W-001	Critical	Waiter logs in with unique Waiter ID
FR-W-002	Critical	Waiter can view assigned tables (maximum 6 tables per waiter)

FR-W-003	Critical	Waiter receives notification when order for their table is COMPLETED
FR-W-004	Critical	Waiter can mark order as DELIVERED when food brought to table
FR-W-005	Critical	Waiter can mark table as AVAILABLE when customer leaves
FR-W-006	High	Waiter can handle customer complaints and escalate to manager
FR-W-007	Medium	Waiter can view customer loyalty status for personalized service
FR-W-008	Low	System tracks average delivery time per waiter
FR-W-009	Low	Waiter can request table cleanup notification

2.5 Manager Requirements

ID	Priority	Requirement Description
FR-M-001	Critical	Manager can approve/deny refund requests
FR-M-002	Critical	Manager can view daily sales report: total orders, total revenue, average order value
FR-M-003	High	Manager can view cashier performance: orders processed, average processing time
FR-M-004	High	Manager can view kitchen performance: average prep time, orders exceeding 30 mins
FR-M-005	Critical	Manager can add/edit/deactivate menu items
FR-M-006	Critical	Manager can set menu item availability (mark as unavailable temporarily)

FR-M-007	High	Manager can view hourly sales breakdown to identify peak hours
FR-M-008	Medium	Manager can view customer loyalty statistics: total members, star distribution
FR-M-009	Medium	Manager can generate weekly/monthly revenue reports
FR-M-010	Medium	Manager can view most popular menu items
FR-M-011	High	Manager can override system rules (e.g., manual discount application)
FR-M-012	High	Manager can view all active orders across all statuses

3. Non-Functional Requirements

3.1 Performance Requirements

ID	Requirement	Target Metric
NFR-P-001	Order creation response time	≤ 3 seconds
NFR-P-002	Concurrent order capacity	50 orders during peak hours
NFR-P-003	Kitchen display refresh rate	≤ 1 second after status update
NFR-P-004	Receipt printing time	≤ 2 seconds after payment
NFR-P-005	Database query response time	≤ 500ms for 95th percentile

NFR-P-006	Payment authorization time	≤ 5 seconds

3.2 Reliability Requirements

ID	Requirement	Target Metric
NFR-R-001	System uptime during business hours	99.5% (10 AM - 10 PM)
NFR-R-002	Database backup frequency	Daily at 2:00 AM
NFR-R-003	Transaction integrity	100% ACID compliance
NFR-R-004	Data loss tolerance	Zero tolerance (RPO = 0)
NFR-R-005	Mean Time To Recovery (MTTR)	≤ 30 minutes

3.3 Security Requirements

ID	Requirement	Implementation
NFR-S-001	Staff authentication	Unique credentials per employee
NFR-S-002	Payment data protection	No storage of full card numbers (PCI-DSS compliance)
NFR-S-003	Customer data privacy	Phone numbers hashed in database
NFR-S-004	Manager action logging	All administrative actions logged with timestamp and user ID
NFR-S-005	Session management	Auto-logout after 30 minutes of inactivity

NFR-S-006	Data encryption	TLS 1.3 for data in transit

3.4 Usability Requirements

ID	Requirement	Target Metric
NFR-U-001	Cashier order entry efficiency	≤ 5 clicks to complete order
NFR-U-002	Kitchen display readability	Readable from 10 feet distance
NFR-U-003	Error message clarity	Plain language, no technical jargon
NFR-U-004	Training time for new staff	≤ 2 hours for basic proficiency
NFR-U-005	Touch-screen compatibility	All interfaces optimized for touch

3.5 Scalability Requirements

ID	Requirement	Target Capacity
NFR-SC-001	Monthly order volume	Support growth to 10,000 orders/month
NFR-SC-002	Menu item capacity	Support up to 100 menu items
NFR-SC-003	Customer database	Support 50,000 loyalty members
NFR-SC-004	Historical data retention	3 years of order history

4. Business Rules

4.1 Order Management Rules

Rule ID	Rule Description
BR-001	Order numbers are sequential per day, format: YYYYMMDD-#### (e.g., 20241208-0001)
BR-002	Orders cannot be voided after payment; only refunds allowed
BR-003	Order modification allowed only if status = 'QUEUED' and within 5 minutes of creation
BR-004	Maximum 30 minutes from order creation to completion (Service Level Agreement)
BR-005	Orders exceeding 30 minutes trigger automatic refund eligibility
BR-006	Minimum order value: $5.00
BR-007	Maximum items per order: 50
BR-008	Order status progression must follow: PENDING → QUEUED → IN_PROCESS → COMPLETED → DELIVERED
BR-009	Cancelled orders must have status = 'QUEUED' and be within 5 minutes of creation

4.2 Loyalty Program Rules

Rule ID	Rule Description
BR-010	1 star earned per completed order (not per dollar amount)
BR-011	Stars never expire

BR-012	Discount tiers: 10★=5% off, 25★=10% off, 50★=15% off, 100★=20% off
BR-013	Only one discount per order
BR-014	Stars deducted immediately when discount applied
BR-015	Phone number is unique identifier for loyalty members (10 digits)
BR-016	Customer order history retained for 1 year
BR-017	Discount applied to subtotal before tax calculation
BR-018	Refunded orders do not earn stars; earned stars removed if order refunded

4.3 Payment Rules

Rule ID	Rule Description
BR-019	Payment is immediate (no "pay later" option)
BR-020	One payment method per order (no split payments between methods)
BR-021	Split payment between customers requires separate orders with optional linkage flag
BR-022	Tax rate: 10% applied to subtotal after discount
BR-023	Cash payments require change calculation if amount tendered exceeds total
BR-024	Card payment authorization code must be stored for reconciliation
BR-025	Card last four digits stored; full card number NEVER stored
BR-026	Failed payment must allow retry with alternative payment method

4.4 Refund Rules

Rule ID	Rule Description
BR-027	Refunds allowed for: excessive wait time (>30min), wrong order, quality issues
BR-028	All refunds require manager approval
BR-029	Refund must be issued to original payment method
BR-030	Refunded orders do not earn stars; stars removed if already credited
BR-031	Partial refunds not allowed; full order refund only
BR-032	Refund reason must be documented (minimum 10 characters)
BR-033	Refunds processed within 3-5 business days for card payments

4.5 Table Management Rules

Rule ID	Rule Description
BR-034	Restaurant has 20 tables numbered 1-20
BR-035	One order per table at a time
BR-036	Tables remain occupied until waiter marks as AVAILABLE
BR-037	Each waiter assigned maximum 6 tables per shift
BR-038	Table assignment is automatic based on availability (lowest available number)
BR-039	Takeout orders have no table assignment

4.6 Kitchen Operations Rules

Rule ID	Rule Description

BR-040	Orders prepared in FIFO (First In, First Out) sequence
BR-041	Chef cannot skip orders in queue without manager override
BR-042	Special instructions must be reviewed before marking order IN_PROCESS
BR-043	Order status progression cannot be reversed (no IN_PROCESS → QUEUED)
BR-044	Chef must update status within 2 minutes of actual state change

4.7 Menu Rules

Rule ID	Rule Description
BR-045	Menu items categorized as: Appetizer, Main, Dessert, Beverage
BR-046	Menu items have estimated prep time (5-20 minutes)
BR-047	Menu items can be marked unavailable (temporary) without deletion
BR-048	Price changes require manager authorization
BR-049	Deleted menu items retained in database for historical order integrity
BR-050	Menu item names must be unique within category

4.8 Staff Rules

Rule ID	Rule Description
BR-051	Each staff member has unique employee ID
BR-052	Staff roles are disjoint (one employee has one primary role)
BR-053	Cashiers and waiters work in shifts (maximum 8 hours)
BR-054	One manager on duty at all times during business hours

BR-055	Staff cannot delete or modify historical data (orders, transactions)
BR-056	Staff credentials expire after 90 days (password change required)

5. Use Case Scenarios

5.1 UC-001: New Customer Places First Order (Dine-In)

Primary Actor: Customer

Secondary Actors: Cashier, Chef, Waiter

Preconditions: Customer is at cashier counter

Postconditions: Order delivered, customer receives food, table available for next customer

Normal Flow:

1. Customer reviews menu board
2. Cashier logs in with Cashier ID (C001)
3. Cashier asks for phone number
4. Customer provides phone number (555-1234)
5. System checks: customer not found, offers loyalty enrollment
6. Customer agrees, system creates loyalty account
7. Customer orders: 1× Burger ($12), 1× Fries ($4), 1× Coke ($3)
8. Customer requests: "No onions on burger"
9. Cashier adds special instruction to Burger item
10. Cashier confirms order details verbally
11. System calculates: Subtotal $19.00, Tax $1.90, Total $20.90
12. Customer chooses dine-in
13. Customer pays $25.00 cash
14. System assigns Table 5, processes payment, generates Order #20241208-0012
15. Customer receives receipt with order number, table number, change $4.10
16. Order sent to kitchen display (status: QUEUED)
17. Customer sits at Table 5

18. Waiter W003 automatically assigned to Table 5
19. Chef marks order IN_PROCESS at 12:05 PM
20. Chef prepares food, following "no onions" instruction
21. Chef marks COMPLETED at 12:18 PM (13 minutes elapsed)
22. Waiter W003 receives notification
23. Waiter delivers food to Table 5, marks DELIVERED
24. Customer eats and leaves
25. Waiter marks Table 5 as AVAILABLE
26. Customer account credited with 1 star (balance: 1 star)

Alternative Flow 1A: Customer already has loyalty account

- At step 5: System finds existing account, displays current stars (e.g., 8 stars)
- Cashier informs: "You have 8 stars, 2 more for a 5% discount!"

Alternative Flow 1B: Customer wants to use discount

- Customer has 12 stars
- At step 11: Cashier applies 10-star discount (5% off)
- New calculation: Subtotal $19.00 - $0.95 discount = $18.05, Tax $1.81, Total $19.86
- Stars deducted: 12 → 2 stars remaining

Exception Flow 1A: All tables occupied

- At step 14: System indicates no tables available
- Cashier suggests takeout or provides wait time estimate (e.g., "15 minutes")
- Customer chooses to wait or switches to takeout

Exception Flow 1B: Order exceeds 30-minute SLA

- At step 21: Order still IN_PROCESS at 12:35 PM (30+ minutes)
- System alerts manager automatically
- Manager offers refund or complimentary dessert
- If refund accepted: Order marked for refund, star removed from customer account

5.2 UC-002: Customer Modifies Order Before Preparation

Primary Actor: Customer

Preconditions: Order placed, payment completed, order status = QUEUED, < 5 minutes elapsed

Normal Flow:

1. Customer (Order #20241208-0015) approaches cashier 2 minutes after ordering
2. Customer: "Can I add a dessert?"
3. Cashier retrieves order #20241208-0015
4. System checks: Order status = QUEUED, created 2 minutes ago (within 5-minute window)
5. Cashier adds 1× Chocolate Cake ($6.00)
6. System recalculates: Original total $20.90 + $6.00 + $0.60 tax = $27.50
7. Additional payment due: $6.60
8. Customer pays $6.60 cash
9. System updates order, generates supplementary receipt
10. Kitchen display refreshes with updated order
11. Chef prepares complete order including cake

Exception Flow 2A: Order already in process

- At step 4: System shows order status = IN_PROCESS
- Cashier: "I'm sorry, the chef has already started preparing. I can create a separate order for the dessert?"
- Customer agrees, new order created (linked to original for table reference)

Exception Flow 2B: Modification requested after 5-minute window

- At step 4: System shows order created 7 minutes ago
- System denies modification based on BR-003
- Cashier explains policy and offers new order option

5.3 UC-003: Manager Approves Refund for Excessive Wait Time

Primary Actor: Manager

Secondary Actors: Cashier, Customer

Preconditions: Order exceeded 30-minute SLA

Normal Flow:

1. Order #20241208-0008 marked COMPLETED at 12:45 PM (created 12:10 PM = 35 minutes elapsed)
2. System automatically flags order for potential refund
3. Customer approaches cashier: "This took way too long!"
4. Cashier: "I sincerely apologize for the delay. Let me get the manager to assist you."
5. Manager M001 reviews order timeline in system
6. Manager confirms: 35 minutes elapsed (exceeds 30-minute SLA per BR-004)
7. Manager initiates refund request in system
8. Manager enters refund details:
    - Order ID: #20241208-0008
    - Refund Amount: $23.50 (full order total)
    - Reason: "Excessive wait time - 35 minutes"
9. Manager approves refund
10. System processes refund to original payment method (Card ending ×××× 4532)
11. System updates order status: DELIVERED → REFUNDED
12. System removes earned star from customer account (if already credited)
13. Customer receives refund confirmation receipt
14. Manager documents incident for kitchen performance review

Alternative Flow 3A: Customer already consumed food

- At step 6: Manager observes food was delivered and consumed
- Manager offers alternative compensation: 50% refund OR free dessert voucher for next visit
- Customer selects voucher option
- System creates promotional code valid for 30 days

Exception Flow 3A: Refund already processed

- At step 7: System shows refund already issued for this order
- Manager informs customer, verifies refund status
- Offers additional compensation if customer remains dissatisfied

5.4 UC-004: Kitchen Handles Multiple Orders During Peak Hours

Primary Actor: Chef

Preconditions: Multiple orders in queue (8+ orders)

Normal Flow:

1. 12:15 PM: Chef views kitchen display showing 8 orders:
    - Order #0020 (12:10) - QUEUED - Status: Green (<15 min)
    - Order #0021 (12:11) - QUEUED - Status: Green
    - Order #0022 (12:12) - QUEUED - Status: Green
    - Order #0023 (12:13) - IN_PROCESS - Status: Yellow (15-25 min)
    - Orders #0024-0027 - QUEUED
2. Chef marks Order #0020 as IN_PROCESS (12:15 PM)
3. Chef reviews order details:
    - Items: 2× Burger, 1× Salad
    - Special instruction: "Gluten-free bun for both burgers"
4. Chef prepares order using gluten-free buns as specified
5. Chef marks #0020 COMPLETED at 12:27 PM (12 minutes prep time)
6. Chef immediately starts Order #0021 (maintaining FIFO sequence)
7. 12:30 PM: Order #0022 indicator changes to Yellow (18 minutes elapsed)
8. 12:35 PM: System alerts - Order #0022 approaching 25 minutes (Yellow → Red threshold)
9. Chef maintains FIFO discipline, completes #0021 at 12:34 PM
10. Chef immediately starts #0022
11. Order #0022 completed at 12:42 PM (30 minutes exactly)
12. System marks order within SLA (≤30 minutes per BR-004) but flags for performance review

Exception Flow 4A: Order exceeds critical threshold

- Order #0024 reaches 31 minutes while chef is busy
- System sends critical alert to manager
- Manager enters kitchen, assists with plating
- Order delivered at 32 minutes
- Manager pre-approves customer compensation

5.5 UC-005: Waiter Manages Multiple Table Deliveries

Primary Actor: Waiter (W002)

Preconditions: Waiter assigned to 6 tables

Normal Flow:

1. 1:00 PM: Waiter W002 logs into system
2. System displays assigned tables:
    - Table 3: OCCUPIED (Order #0045 in progress)
    - Table 7: AVAILABLE
    - Table 9: OCCUPIED (Order #0047 in progress)
    - Table 12: AVAILABLE
    - Table 15: OCCUPIED (Order #0048 in progress)
    - Table 18: AVAILABLE
3. 1:05 PM: Order #0045 (Table 3) marked COMPLETED by chef
4. Waiter receives notification: "Order #0045 ready - Table 3"
5. Waiter retrieves order from kitchen pass
6. Waiter delivers to Table 3, marks status DELIVERED
7. Waiter: "Enjoy your meal! Please let me know if you need anything."
8. 1:08 PM: Notification received for Table 15 (Order #0048)
9. Waiter delivers to Table 15, marks DELIVERED
10. 1:15 PM: Customer at Table 3 leaves
11. Waiter clears table, marks Table 3 as AVAILABLE in system
12. 1:16 PM: New customer assigned Table 3 (Order #0052 created)
13. Cycle continues throughout shift

Alternative Flow 5A: Customer complaint handling

- At step 7: Customer: "This burger is cold"
- Waiter: "I sincerely apologize. Let me get you a fresh burger right away."
- Waiter escalates issue to manager via system
- Manager reviews and approves remake
- Chef prioritizes remake order
- Fresh burger delivered within 8 minutes
- Waiter updates order notes with resolution

6. Entity Identification

This section presents the finalized entity model derived from the requirements analysis. Each entity represents a distinct concept with independent existence and attributes.

6.1 Core Entities

6.1.1 CUSTOMER

Definition: Represents individuals who place orders at the restaurant. Customers may be anonymous (guest) or registered (loyalty members).

Rationale: Customers have independent existence and persist across multiple orders. Loyalty program requires tracking customer history and rewards.

Primary Key: `customer_id` (Surrogate key)

Candidate Keys:

- `phone_number` (Natural key, unique, nullable for guest customers)

Attributes:

- `customer_id` (INT, PK, Auto-increment)
- `phone_number` (VARCHAR(15), Unique, Nullable)
- `phone_number_hash` (CHAR(64), Indexed) - For privacy compliance
- `first_name` (VARCHAR(50), Nullable)
- `last_name` (VARCHAR(50), Nullable)
- `total_stars` (INT, Default: 0) - Current loyalty star balance
- `total_lifetime_orders` (INT, Default: 0) - Historical order count
- `registration_date` (DATE)
- `last_order_date` (DATE, Nullable)
- `is_vip` (BOOLEAN, Default: FALSE) - Computed: total_stars ≥ 100
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- `updated_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP ON UPDATE)

Business Rules:

- Guest customers have NULL phone_number
- Phone number must be exactly 10 digits (validated at application layer)
- VIP status automatically updated when total_stars ≥ 100
- Customer records are never deleted (soft delete for GDPR compliance)

6.1.2 EMPLOYEE (Supertype)

Definition: Represents all staff members who interact with the DineFlow system. Serves as parent entity for role-based specialization.

Rationale: Common attributes exist across all employee types (name, contact, hire date). Specialization allows role-specific attributes and behaviors.

Primary Key: `employee_id` (Surrogate key)

Specialization Type: DISJOINT (An employee has exactly one primary role)

Specialization Coverage: TOTAL (Every employee must have a role)

Attributes:

- `employee_id` (INT, PK, Auto-increment)
- `first_name` (VARCHAR(50), NOT NULL)
- `last_name` (VARCHAR(50), NOT NULL)
- `phone_number` (VARCHAR(15), Unique, NOT NULL)
- `email` (VARCHAR(100), Unique, NOT NULL)
- `hire_date` (DATE, NOT NULL)
- `employment_status` (ENUM: 'ACTIVE', 'ON_LEAVE', 'TERMINATED', Default: 'ACTIVE')
- `password_hash` (CHAR(64), NOT NULL) - SHA-256 hashed password
- `last_login` (TIMESTAMP, Nullable)
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- `updated_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP ON UPDATE)

Subtypes: CASHIER, CHEF, WAITER, MANAGER

Discriminator Attribute: `employee_role` (ENUM: 'CASHIER', 'CHEF', 'WAITER', 'MANAGER')

Business Rules:

- Each employee assigned exactly one primary role (disjoint specialization)
- Employee ID format: E#### (e.g., E0001)
- Passwords must be changed every 90 days
- Terminated employees retained for historical data integrity

1.1 CASHIER (Subtype of EMPLOYEE)

Definition: Front-line staff responsible for order entry and payment processing.

Additional Attributes:

- `cashier_id` (INT, PK, FK references EMPLOYEE.employee_id)
- `shift_start_time` (TIME, Nullable)
- `shift_end_time` (TIME, Nullable)
- `total_orders_processed` (INT, Default: 0)
- `average_order_processing_time` (DECIMAL(5,2), Nullable) - In minutes
- `last_shift_date` (DATE, Nullable)

Business Rules:

- Must log in at shift start
- Maximum 8-hour shift duration
- Performance metrics updated nightly

1.2 CHEF (Subtype of EMPLOYEE)

Definition: Kitchen staff responsible for food preparation and order fulfillment.

Additional Attributes:

- `chef_id` (INT, PK, FK references EMPLOYEE.employee_id)
- `specialty` (ENUM: 'GRILL', 'FRYER', 'ASSEMBLY', 'DESSERT', 'GENERAL', Default: 'GENERAL')
- `total_orders_prepared` (INT, Default: 0)
- `average_prep_time` (DECIMAL(5,2), Nullable) - In minutes
- `orders_exceeding_sla` (INT, Default: 0) - Orders > 30 minutes

Business Rules:

- Must follow FIFO order queue
- Cannot skip orders without manager override
- Performance tracked for quality assurance

1.3 WAITER (Subtype of EMPLOYEE)

Definition: Service staff responsible for order delivery and table management.

Additional Attributes:

- `waiter_id` (INT, PK, FK references EMPLOYEE.employee_id)
- `max_tables` (INT, Default: 6) - Maximum concurrent table assignments
- `current_table_count` (INT, Default: 0)
- `total_deliveries` (INT, Default: 0)
- `average_delivery_time` (DECIMAL(5,2), Nullable) - In minutes

Business Rules:

- Maximum 6 tables assigned simultaneously
- Must mark tables available after customer departure
- Performance metrics updated in real-time

1.4 MANAGER (Subtype of EMPLOYEE)

Definition: Supervisory staff with administrative and override capabilities.

Additional Attributes:

- `manager_id` (INT, PK, FK references EMPLOYEE.employee_id)
- `authorization_level` (ENUM: 'SHIFT_MANAGER', 'GENERAL_MANAGER', 'OWNER', Default: 'SHIFT_MANAGER')
- `total_refunds_approved` (INT, Default: 0)
- `total_refunds_denied` (INT, Default: 0)

Business Rules:

- At least one manager on duty during business hours
- All refunds require manager approval
- Can override system constraints with logged justification

2. MENU_ITEM

Definition: Represents food and beverage items available for purchase.

Rationale: Menu items have independent existence with attributes like price, description, and preparation time. Items persist even when temporarily unavailable.

Primary Key: `item_id` (Surrogate key)

Attributes:

- `item_id` (INT, PK, Auto-increment)
- `item_name` (VARCHAR(100), NOT NULL, Unique within category)
- `description` (TEXT, Nullable)
- `category` (ENUM: 'APPETIZER', 'MAIN', 'DESSERT', 'BEVERAGE', NOT NULL)
- `price` (DECIMAL(10,2), NOT NULL) - Current selling price
- `cost` (DECIMAL(10,2), Nullable) - Cost of goods (for margin analysis)
- `prep_time_minutes` (INT, NOT NULL) - Estimated preparation time (5-20)
- `is_available` (BOOLEAN, Default: TRUE) - Current availability status
- `is_active` (BOOLEAN, Default: TRUE) - Soft delete flag
- `image_url` (VARCHAR(255), Nullable) - Menu board image
- `allergen_info` (VARCHAR(255), Nullable) - e.g., "Contains nuts, dairy"

- `calories` (INT, Nullable)
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- `updated_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP ON UPDATE)
- `created_by` (INT, FK references MANAGER.manager_id)

Business Rules:

- Price must be > 0
- Prep time between 5-20 minutes
- Deleted items marked inactive (is_active = FALSE) but not removed
- Name must be unique within category
- Price changes logged in separate audit table (future enhancement)

Sample Data:

- Burger (Main, $12.00, 12 min prep)
- Fries (Appetizer, $4.00, 8 min prep)
- Chocolate Cake (Dessert, $6.00, 5 min prep)

3. ORDER

Definition: Represents a customer's purchase request, tracking from creation through fulfillment.

Rationale: Orders are the central transaction entity, linking customers, menu items, payments, and fulfillment workflows.

Primary Key: `order_id` (Surrogate key)

Candidate Keys:

- `order_number` (Business key, unique per day)

Attributes:

- `order_id` (INT, PK, Auto-increment)

- `order_number` (VARCHAR(20), NOT NULL, Unique) - Format: YYYYMMDD-#### (e.g., 20241208-0001)
- `customer_id` (INT, Nullable, FK references CUSTOMER.customer_id) - NULL for guest orders
- `cashier_id` (INT, NOT NULL, FK references CASHIER.cashier_id)
- `chef_id` (INT, Nullable, FK references CHEF.chef_id) - Assigned when cooking starts
- `waiter_id` (INT, Nullable, FK references WAITER.waiter_id) - Assigned for dine-in
- `table_id` (INT, Nullable, FK references TABLE.table_id) - NULL for takeout
- `order_date` (DATE, NOT NULL)
- `order_time` (TIME, NOT NULL)
- `order_type` (ENUM: 'DINE_IN', 'TAKEOUT', NOT NULL)
- `status` (ENUM: 'PENDING', 'QUEUED', 'IN_PROCESS', 'COMPLETED', 'DELIVERED', 'CANCELLED', 'REFUNDED', Default: 'PENDING')
- `status_updated_at` (TIMESTAMP) - Last status change timestamp
- `special_instructions` (TEXT, Nullable) - Order-level instructions (max 500 chars)
- `estimated_wait_time` (INT, Default: 30) - In minutes
- `actual_prep_time` (INT, Nullable) - Calculated: completed_time - queued_time
- `subtotal` (DECIMAL(10,2), NOT NULL)
- `discount_amount` (DECIMAL(10,2), Default: 0.00)
- `tax_amount` (DECIMAL(10,2), NOT NULL)
- `total_amount` (DECIMAL(10,2), NOT NULL)
- `stars_earned` (INT, Default: 1) - Stars credited for this order
- `stars_redeemed` (INT, Default: 0) - Stars used for discount
- `is_flagged_for_review` (BOOLEAN, Default: FALSE) - Exceeds SLA or has issues
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- `queued_at` (TIMESTAMP, Nullable) - When sent to kitchen
- `in_process_at` (TIMESTAMP, Nullable) - When chef started cooking
- `completed_at` (TIMESTAMP, Nullable) - When chef finished
- `delivered_at` (TIMESTAMP, Nullable) - When waiter delivered

- `related_order_id` (INT, Nullable, FK references ORDER.order_id) - For split payment linking

Computed Fields (Application Layer):

- `is_sla_exceeded`: (completed_at - created_at) > 30 minutes
- `display_color`: Green (<15min), Yellow (15-25min), Red (>25min)

Business Rules:

- Order number resets daily at midnight
- Status must progress sequentially (no backwards transitions)
- DINE_IN orders must have table_id
- TAKEOUT orders must have table_id = NULL
- Discount applied before tax calculation: tax = (subtotal - discount) × 0.10
- Orders older than 1 year archived to separate table

Status Lifecycle:

1. **PENDING**: Order being entered by cashier (pre-payment)
2. **QUEUED**: Payment complete, waiting for chef to start
3. **IN_PROCESS**: Chef actively preparing
4. **COMPLETED**: Food ready, waiting for delivery
5. **DELIVERED**: Food delivered to customer (dine-in) or picked up (takeout)
6. **CANCELLED**: Order cancelled before preparation (< 5 min, pre-queue)
7. **REFUNDED**: Order refunded after completion

4. ORDER_ITEM (Associative Entity)

Definition: Represents the many-to-many relationship between ORDERs and MENU_ITEMs, capturing line-item details.

Rationale: This is NOT just a relationship—it has attributes (quantity, price snapshot, item-level instructions). Associative entity required.

Primary Key: Composite (`order_id`, `item_id`, `line_number`)

Attributes:

- `order_id` (INT, PK, FK references ORDER.order_id)
- `item_id` (INT, PK, FK references MENU_ITEM.item_id)
- `line_number` (INT, PK) - Sequence within order (1, 2, 3...)
- `quantity` (INT, NOT NULL) - Range: 1-99
- `unit_price` (DECIMAL(10,2), NOT NULL) - Price snapshot at order time
- `special_instructions` (VARCHAR(200), Nullable) - Item-specific (e.g., "No onions")
- `subtotal` (DECIMAL(10,2), NOT NULL) - quantity × unit_price
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)

Business Rules:

- unit_price captured at order time (historical accuracy if menu price changes)
- Quantity must be 1-99
- Line numbers sequential within order
- Cannot be deleted after order status = QUEUED

Sample Record:

- Order #20241208-0001, Burger (item_id=5), line 1, qty=2, price=$12.00, subtotal=$24.00, instructions="No onions"

5. TRANSACTION

Definition: Financial record of payment for an order. One-to-one relationship with ORDER.

Rationale: Separate entity for financial audit trail and reconciliation. Payment details independent from order fulfillment.

Primary Key: `transaction_id` (Surrogate key)

Attributes:

- `transaction_id` (INT, PK, Auto-increment)

- `order_id` (INT, NOT NULL, Unique, FK references ORDER.order_id) - One transaction per order
- `transaction_date` (DATE, NOT NULL)
- `transaction_time` (TIME, NOT NULL)
- `payment_method` (ENUM: 'CASH', 'CREDIT', 'DEBIT', NOT NULL)
- `subtotal` (DECIMAL(10,2), NOT NULL) - Before discount and tax
- `discount_amount` (DECIMAL(10,2), Default: 0.00)
- `tax_amount` (DECIMAL(10,2), NOT NULL)
- `total_amount` (DECIMAL(10,2), NOT NULL) - Final amount charged
- `amount_tendered` (DECIMAL(10,2), Nullable) - For cash payments only
- `change_amount` (DECIMAL(10,2), Nullable) - For cash payments only
- `card_last_four` (CHAR(4), Nullable) - Last 4 digits for card payments
- `card_authorization_code` (VARCHAR(20), Nullable) - Bank auth code
- `card_type` (ENUM: 'VISA', 'MASTERCARD', 'AMEX', 'DISCOVER', Nullable)
- `transaction_status` (ENUM: 'PENDING', 'COMPLETED', 'FAILED', 'REFUNDED', Default: 'PENDING')
- `processed_by` (INT, NOT NULL, FK references CASHIER.cashier_id)
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)

Business Rules:

- One transaction per order (one-to-one relationship)
- CASH: amount_tendered and change_amount required
- CARD: card_last_four and authorization_code required
- NEVER store full card number (PCI-DSS compliance)
- Total amount must match order total
- Failed transactions do not create orders

Payment Method Constraints:

```
CONSTRAINT chk_cash_payment
  CHECK (payment_method != 'CASH' OR (amount_tendered IS NOT NULL AND
change_amount IS NOT NULL))
```

CONSTRAINT chk_card_payment
  CHECK (payment_method = 'CASH' OR (card_last_four IS NOT NULL AND card_authorization_code IS NOT NULL))

6. TABLE

Definition: Physical tables in the restaurant available for dine-in customers.

Rationale: Tables are physical resources requiring allocation, tracking, and status management.

Primary Key: `table_id` (Surrogate key)

Candidate Keys:

- `table_number` (Business key, unique)

Attributes:

- `table_id` (INT, PK, Auto-increment)
- `table_number` (INT, NOT NULL, Unique) - Range: 1-20
- `capacity` (INT, NOT NULL) - Number of seats (2, 4, 6)
- `is_available` (BOOLEAN, Default: TRUE)
- `current_order_id` (INT, Nullable, FK references ORDER.order_id) - NULL when available
- `assigned_waiter_id` (INT, Nullable, FK references WAITER.waiter_id)
- `occupied_since` (TIMESTAMP, Nullable) - When customer sat down
- `last_cleaned` (TIMESTAMP, Nullable)
- `location_zone` (ENUM: 'FRONT', 'MIDDLE', 'BACK', 'PATIO', Default: 'MIDDLE')
- `is_active` (BOOLEAN, Default: TRUE) - For maintenance/removal

Business Rules:

- Table numbers 1-20 (20 tables total per BR-034)

- One order per table at a time (BR-035)
- Table remains occupied until waiter marks available (BR-036)
- Each waiter manages max 6 tables (enforced at application layer)
- Automatic assignment: lowest available table number (BR-038)

Table Lifecycle:

1. **AVAILABLE**: No current order, ready for assignment
2. **OCCUPIED**: Has current_order_id, customer seated
3. **NEEDS_CLEANING**: Customer left, awaiting cleanup
4. **MAINTENANCE**: Temporarily unavailable

7. REFUND

Definition: Record of financial refund issued for an order. Requires manager approval.

Rationale: Separate entity for audit trail, financial reconciliation, and managerial oversight.

Primary Key: `refund_id` (Surrogate key)

Attributes:

- `refund_id` (INT, PK, Auto-increment)
- `order_id` (INT, NOT NULL, Unique, FK references ORDER.order_id) - One refund per order
- `transaction_id` (INT, NOT NULL, FK references TRANSACTION.transaction_id)
- `refund_date` (DATE, NOT NULL)
- `refund_time` (TIME, NOT NULL)
- `refund_amount` (DECIMAL(10,2), NOT NULL) - Must equal original transaction total
- `refund_reason` (TEXT, NOT NULL) - Minimum 10 characters (BR-032)
- `reason_category` (ENUM: 'EXCESSIVE_WAIT', 'WRONG_ORDER', 'QUALITY_ISSUE', 'CUSTOMER_REQUEST', 'OTHER')
- `requested_by` (INT, NOT NULL, FK references CASHIER.cashier_id)
- `approved_by` (INT, NOT NULL, FK references MANAGER.manager_id)

- `approval_status` (ENUM: 'PENDING', 'APPROVED', 'DENIED', Default: 'PENDING')
- `approval_date` (DATE, Nullable)
- `refund_method` (ENUM: 'ORIGINAL_PAYMENT', 'CASH', 'STORE_CREDIT') - Must match original (BR-029)
- `stars_reversed` (INT, Default: 0) - Stars removed from customer account
- `manager_notes` (TEXT, Nullable)
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)
- `processed_at` (TIMESTAMP, Nullable)

Business Rules:

- All refunds require manager approval (BR-028)
- Refund amount must equal original transaction total (BR-031)
- Stars earned from order must be reversed (BR-030)
- Refund issued to original payment method (BR-029)
- Reason required, minimum 10 characters (BR-032)

Refund Workflow:

1. Cashier initiates refund request
2. Manager reviews and approves/denies
3. If approved: transaction processed, order status → REFUNDED
4. Customer account updated: stars removed, refund recorded

8. LOYALTY_STAR_TRANSACTION (Audit Entity)

Definition: Historical record of all star earning and redemption events for loyalty customers.

Rationale: Provides audit trail for customer disputes, compliance, and analytics. Customer table stores only current balance.

Primary Key: `star_transaction_id` (Surrogate key)

Attributes:

- `star_transaction_id` (INT, PK, Auto-increment)

- `customer_id` (INT, NOT NULL, FK references CUSTOMER.customer_id)
- `transaction_type` (ENUM: 'EARNED', 'REDEEMED', 'REVERSED', 'ADJUSTED', NOT NULL)
- `star_amount` (INT, NOT NULL) - Positive for earned, negative for redeemed
- `balance_before` (INT, NOT NULL) - Star balance before this transaction
- `balance_after` (INT, NOT NULL) - Star balance after this transaction
- `order_id` (INT, Nullable, FK references ORDER.order_id) - Linked order if applicable
- `transaction_id` (INT, Nullable, FK references TRANSACTION.transaction_id) - Linked payment
- `refund_id` (INT, Nullable, FK references REFUND.refund_id) - If reversed due to refund
- `discount_percentage` (DECIMAL(5,2), Nullable) - If redeemed (5.00, 10.00, 15.00, 20.00)
- `transaction_date` (DATE, NOT NULL)
- `transaction_time` (TIME, NOT NULL)
- `notes` (TEXT, Nullable)
- `created_by` (INT, Nullable, FK references EMPLOYEE.employee_id) - For manual adjustments
- `created_at` (TIMESTAMP, Default: CURRENT_TIMESTAMP)

Business Rules:

- Every star change creates a transaction record
- balance_after = balance_before + star_amount
- EARNED: +1 star per order (BR-010)
- REDEEMED: -10, -25, -50, or -100 stars for discounts (BR-012)
- REVERSED: Negative transaction when order refunded (BR-018)
- Transactions never deleted (immutable audit log)

Sample Records:

Transaction 1: Customer 123, EARNED, +1, balance 0→1, Order #0001
Transaction 2: Customer 123, EARNED, +1, balance 1→2, Order #0015

Transaction 3: Customer 123, REDEEMED, -10, balance 12→2, Order #0023 (5% discount)
Transaction 4: Customer 123, REVERSED, -1, balance 2→1, Refund #5 (reversed Order #0023)

9. SYSTEM ENTITIES (Supporting/Reference Data)

9.1 AUDIT_LOG

Definition: System-wide audit trail for security and compliance.

Attributes:

- `log_id` (BIGINT, PK, Auto-increment)
- `log_type` (ENUM: 'LOGIN', 'LOGOUT', 'REFUND', 'MENU_CHANGE', 'OVERRIDE', 'ERROR')
- `employee_id` (INT, Nullable, FK references EMPLOYEE.employee_id)
- `action_description` (TEXT, NOT NULL)
- `table_affected` (VARCHAR(50), Nullable)
- `record_id` (INT, Nullable)
- `ip_address` (VARCHAR(45), Nullable)
- `timestamp` (TIMESTAMP, Default: CURRENT_TIMESTAMP)

Business Rules:

- All manager actions logged (BR-055, NFR-S-004)
- Logs retained for 7 years (compliance)
- No updates or deletes allowed (immutable)

9.2 SYSTEM_CONFIG

Definition: System-wide configuration parameters.

Attributes:

- `config_key` (VARCHAR(50), PK)

- `config_value` (TEXT, NOT NULL)
- `data_type` (ENUM: 'STRING', 'INTEGER', 'DECIMAL', 'BOOLEAN')
- `description` (TEXT)
- `updated_by` (INT, FK references MANAGER.manager_id)
- `updated_at` (TIMESTAMP)

Sample Configurations:

- `tax_rate`: 0.10 (10%)
- `order_sla_minutes`: 30
- `max_items_per_order`: 50
- `min_order_value`: 5.00
- `max_waiter_tables`: 6

10. Entity Relationship Summary

Entity	Type	Key Relationships
CUSTOMER	Strong	1:N with ORDER, 1:N with LOYALTY_STAR_TRANSACTION
EMPLOYEE	Strong (Supertype)	Specialized into CASHIER, CHEF, WAITER, MANAGER
CASHIER	Subtype	1:N with ORDER (creates), 1:N with TRANSACTION (processes)
CHEF	Subtype	1:N with ORDER (prepares)
WAITER	Subtype	1:N with ORDER (delivers), 1:N with TABLE (manages)
MANAGER	Subtype	1:N with REFUND (approves), 1:N with MENU_ITEM (manages)

MENU_ITEM	Strong	M:N with ORDER (via ORDER_ITEM)
ORDER	Strong	N:1 with CUSTOMER, 1:N with ORDER_ITEM, 1:1 with TRANSACTION, N:1 with TABLE
ORDER_ITEM	Associative	Links ORDER and MENU_ITEM with attributes
TRANSACTION	Strong	1:1 with ORDER, N:1 with CASHIER
TABLE	Strong	1:1 with ORDER (when occupied), N:1 with WAITER
REFUND	Strong	1:1 with ORDER, 1:1 with TRANSACTION, N:1 with MANAGER
LOYALTY_STAR_TRANSACTION	Weak	N:1 with CUSTOMER, N:1 with ORDER (optional)
AUDIT_LOG	Strong	N:1 with EMPLOYEE (optional)
SYSTEM_CONFIG	Strong	Configuration reference table

11. Cardinality and Participation Constraints

11.1 CUSTOMER Relationships

CUSTOMER places ORDER

- **Cardinality:** 1:N (One customer can place many orders)
- **Participation:**
    - CUSTOMER: Partial (customers exist before placing orders)
    - ORDER: Partial (guest orders have no customer_id)
- **Business Rule:** Guest customers (NULL customer_id) allowed per BR-015

CUSTOMER has LOYALTY_STAR_TRANSACTION

- **Cardinality:** 1:N (One customer has many star transactions)
- **Participation:**
    - CUSTOMER: Partial (new customers have no transactions yet)
    - LOYALTY_STAR_TRANSACTION: Total (every transaction must belong to a customer)

11.2 EMPLOYEE Relationships

CASHIER creates ORDER

- **Cardinality:** 1:N (One cashier creates many orders)
- **Participation:**
    - CASHIER: Partial (new cashiers haven't created orders yet)
    - ORDER: Total (every order must be created by a cashier)

CASHIER processes TRANSACTION

- **Cardinality:** 1:N
- **Participation:** Both Total (every transaction processed by cashier, cashiers process transactions)

CHEF prepares ORDER

- **Cardinality:** 1:N
- **Participation:**
    - CHEF: Partial (assigned when order moves to IN_PROCESS)
    - ORDER: Partial (order created before chef assignment)

WAITER delivers ORDER

- **Cardinality:** 1:N (One waiter delivers many orders)
- **Participation:**
    - WAITER: Partial
    - ORDER: Partial (only DINE_IN orders have waiters)

WAITER manages TABLE

- **Cardinality:** 1:N (One waiter manages up to 6 tables)
- **Participation:**
    - WAITER: Partial (waiters may have no tables during slow periods)
    - TABLE: Partial (tables may be unassigned)

MANAGER approves REFUND

- **Cardinality:** 1:N (One manager approves many refunds)
- **Participation:**
    - MANAGER: Partial
    - REFUND: Total (every refund requires manager approval)

MANAGER manages MENU_ITEM

- **Cardinality:** 1:N (One manager creates/modifies many items)
- **Participation:** Both Partial

11.3 ORDER Relationships

ORDER contains MENU_ITEM (via ORDER_ITEM)

- **Cardinality:** M:N (Many orders contain many menu items)
- **Participation:** Both Total (orders must have items, items must be in orders)
- **Associative Entity:** ORDER_ITEM with attributes

ORDER has TRANSACTION

- **Cardinality:** 1:1 (One order has exactly one transaction)
- **Participation:**
    - ORDER: Total (every order must be paid)
    - TRANSACTION: Total (every transaction for exactly one order)

ORDER occupies TABLE

- **Cardinality:** N:1 (Many orders can occupy one table over time, but one order per table at a time)
- **Participation:**
    - ORDER: Partial (only DINE_IN orders)

    - TABLE: Partial (tables available when no order)

ORDER has REFUND

- **Cardinality:** 1:1 (One order can have at most one refund)
- **Participation:** Both Partial (not all orders refunded)

12. Weak Entity Analysis

LOYALTY_STAR_TRANSACTION is a **weak entity**:

- **Owner Entity:** CUSTOMER
- **Identifying Relationship:** "has"
- **Partial Key:** `star_transaction_id` (becomes unique when combined with customer_id)
- **Rationale:** Star transactions have no meaning without a customer context. If customer deleted (hypothetically), all star transactions would be removed.

All other entities are strong entities with independent existence.

13. Specialization/Generalization Details

EMPLOYEE Hierarchy

Supertype: EMPLOYEE

Specialization Characteristics:

- **Type:** Disjoint (∂) - An employee has exactly ONE primary role
- **Coverage:** Total - Every employee MUST have a role
- **Discriminator:** `employee_role` attribute

14. Data Integrity Constraints

14.1 Entity-Level Constraints

CUSTOMER:

CHECK (phone_number IS NULL OR LENGTH(phone_number) = 10)
CHECK (total_stars >= 0)

ORDER:

CHECK (total_amount = subtotal - discount_amount + tax_amount)
CHECK (tax_amount = (subtotal - discount_amount) * 0.10)
CHECK (order_type = 'TAKEOUT' OR table_id IS NOT NULL)
CHECK (estimated_wait_time > 0 AND estimated_wait_time <= 60)
CHECK (status IN ('PENDING', 'QUEUED', 'IN_PROCESS', 'COMPLETED', 'DELIVERED', 'CANCELLED', 'REFUNDED'))

ORDER_ITEM:

CHECK (quantity > 0 AND quantity <= 99)
CHECK (subtotal = quantity * unit_price)
CHECK (LENGTH(special_instructions) <= 200)

TRANSACTION:

CHECK (total_amount > 0)
CHECK (payment_method = 'CASH' OR amount_tendered IS NULL)
CHECK (payment_method != 'CASH' OR amount_tendered >= total_amount)
CHECK (change_amount = amount_tendered - total_amount OR change_amount IS NULL)

MENU_ITEM:

CHECK (price > 0)
CHECK (prep_time_minutes >= 5 AND prep_time_minutes <= 20)

TABLE:

CHECK (table_number >= 1 AND table_number <= 20)
CHECK (capacity IN (2, 4, 6))

REFUND:

CHECK (LENGTH(refund_reason) >= 10)
CHECK (approval_status IN ('PENDING', 'APPROVED', 'DENIED'))