We will delve into each model’s purpose, attributes, relationships, and significance within the overall system architecture. This comprehensive explanation aims to provide a thorough understanding of the data structure underpinning your WMS.

**1. Address Model**

* **Purpose:** The Address model is designed to store and manage address information for various entities within the WMS, such as suppliers, customers (though not explicitly defined in your models, it’s a common association), and potentially even the warehouse itself. This model promotes data normalization by centralizing address details, preventing redundancy and ensuring consistency across the system.
* **Attributes:**
  + AddressID (INTEGER, primaryKey, autoIncrement): This attribute serves as the unique identifier for each address record. It’s an integer that automatically increments with each new address created, ensuring that every address has a distinct and easily referenceable ID. As the primary key, it’s the fundamental way to access and relate to specific address records in other tables.
  + AddressLine1 (STRING(255), allowNull: false): This field stores the primary address line, typically including the street number and name. The STRING(255) data type indicates that it can hold a string of up to 255 characters, providing ample space for most address formats. The allowNull: false constraint mandates that this field must contain a value when a new address is created, as it’s a fundamental part of any address.
  + AddressLine2 (STRING(255)): This optional field allows for a secondary address line, which might include apartment numbers, suite numbers, or other additional address details. The absence of allowNull: false indicates that this field can be left empty if not applicable.
  + City (STRING(100), allowNull: false): This attribute stores the city component of the address. The STRING(100) type allows for city names up to 100 characters. The allowNull: false constraint ensures that a city must be specified for every address.
  + StateProvince (STRING(100)): This optional field holds the state or province information for the address. Like AddressLine2, its optional nature (no allowNull: false) allows for flexibility in handling addresses from regions where this information might not be required or applicable.
  + PostalCode (STRING(20), allowNull: false): This field stores the postal code or zip code of the address. The STRING(20) type accommodates various postal code formats used internationally. The allowNull: false constraint ensures that a postal code is provided, as it’s crucial for mail and shipping purposes.
  + Country (STRING(100), allowNull: false): This attribute specifies the country of the address. The STRING(100) type allows for country names up to 100 characters. The allowNull: false constraint makes it a mandatory field for every address record.
  + AddressType (STRING(50)): This optional field can be used to categorize the type of address, such as ‘Shipping’, ‘Billing’, or ‘Warehouse’. This allows for a single Address model to be used in different contexts, with the AddressType helping to distinguish the purpose of a particular address record.
* **Relationships:** The Address model has the potential to have relationships with several other models in the system. Based on your other model definitions:
  + Supplier: The Supplier model has a foreign key AddressID referencing the Address model, indicating a one-to-one or one-to-many relationship where a supplier has one main address.
  + OutboundOrder: The OutboundOrder model has foreign keys ShippingAddressID and BillingAddressID referencing the Address model, indicating that each outbound order has associated shipping and billing addresses.
  + PurchaseOrder: Similar to OutboundOrder, the PurchaseOrder model also has foreign keys ShippingAddressID and BillingAddressID referencing the Address model.
* **Indexes:** There are no explicit indexes defined in the Address model schema provided. However, you might consider adding indexes on frequently queried fields like PostalCode or Country to improve the performance of address-based searches.
* **Timestamps:** The timestamps: false configuration indicates that this model does not automatically manage createdAt and updatedAt timestamps, which are typically used to track when a record was created and last updated.
* **Table Name:** The tableName: 'addresses' configuration specifies that the database table associated with this model is named ‘addresses’.
* **Example Use Cases:**
  + Storing the primary business address of a supplier.
  + Recording the shipping and billing addresses for a customer placing an order.
  + Potentially storing the physical location of different warehouses or storage facilities.

**2. Inventory Model**

* **Purpose:** The Inventory model is central to your WMS, responsible for tracking the quantity of each product in specific locations within the warehouse. It links products (via ProductLot) to physical locations and maintains information about the available and allocated quantities.
* **Attributes:**
  + InventoryID (INTEGER, primaryKey, autoIncrement): The unique identifier for each inventory record, automatically incrementing for new entries.
  + ProductLotID (INTEGER, allowNull: false, foreign key referencing ProductLots): This is a crucial foreign key that links the inventory record to a specific lot of a product. The allowNull: false constraint ensures that every inventory record is associated with a product lot. The references property establishes the relationship with the ProductLots table via the ProductLotID key.
  + LocationID (INTEGER, allowNull: false, foreign key referencing WarehouseLocations): This foreign key links the inventory record to a specific location within the warehouse. The allowNull: false constraint mandates that every inventory record must be associated with a location. The references property defines the relationship with the WarehouseLocations table using the LocationID key.
  + QuantityOnHand (INTEGER, allowNull: false, defaultValue: 0): This attribute stores the current physical quantity of the product lot available at the specified location. The allowNull: false constraint and defaultValue: 0 ensure that the quantity on hand is always a non-null integer, starting at zero if not explicitly set.
  + QuantityAllocated (INTEGER, defaultValue: 0): This field tracks the quantity of the product lot at this location that has been allocated to fulfill outbound orders but has not yet been shipped. The defaultValue: 0 indicates that initially, no quantity is allocated.
  + SerialNumber (STRING): This optional field can store the serial number of the product, if applicable. This is particularly useful for tracking individual items that require serial number management.
  + LastStockTakeDate (DATE): This field records the date of the last stock take for this specific inventory record. This information is valuable for audit purposes and for managing inventory accuracy.
  + Notes (TEXT): This optional field allows for any additional notes or information related to this inventory record.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): This attribute automatically records the timestamp when the inventory record was created. The defaultValue: DataTypes.NOW ensures that the current date and time are captured upon creation.
* **Relationships:**
  + belongsTo ProductLot (foreignKey: ProductLotID): Each inventory record belongs to a specific product lot.
  + belongsTo WarehouseLocation (foreignKey: LocationID): Each inventory record is located in a specific warehouse location.
* **Indexes:**
  + unique index on ['ProductLotID', 'LocationID']: This crucial index ensures that there is only one inventory record for a specific product lot at a given location. This prevents inconsistencies and ensures data integrity.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps, providing a history of when the record was created and last modified.
* **Table Name:** The tableName: 'inventories' configuration specifies the database table name.
* **Example Use Cases:**
  + Tracking that there are 10 units of Product ‘A’, Lot ‘123’, located in Warehouse Location ‘Aisle-01-Shelf-05’.
  + Showing that 2 of those 10 units are currently allocated to an open outbound order.
  + Recording the last time a physical count was performed for that specific inventory.

**3. OutboundOrderLineItem Model**

* **Purpose:** The OutboundOrderLineItem model represents a single item within an outbound order. It details the specific product, the quantity ordered, the quantity shipped, and other relevant information for that line item.
* **Attributes:**
  + OrderItemID (INTEGER, primaryKey, autoIncrement): The unique identifier for each line item within an outbound order.
  + OrderID (INTEGER, allowNull: false, foreign key referencing OutboundOrders, onDelete: 'CASCADE'): This foreign key links the line item to the specific outbound order it belongs to. The allowNull: false constraint ensures that every line item is associated with an order. The onDelete: 'CASCADE' setting means that if an outbound order is deleted, all its associated line items will also be automatically deleted.
  + ProductID (INTEGER, allowNull: false, foreign key referencing Products): This foreign key identifies the specific product included in this line item. The allowNull: false constraint ensures that a product is specified for each line item.
  + QuantityOrdered (INTEGER, allowNull: false): This attribute stores the initial quantity of the product that the customer ordered in this line item.
  + QuantityShipped (INTEGER, defaultValue: 0): This field tracks the actual quantity of the product that has been shipped for this line item. It’s initialized to zero and updated as shipments are processed.
  + UnitOfMeasure (STRING(50), allowNull: false): This attribute specifies the unit of measure in which the product was ordered (e.g., ‘Each’, ‘Box’, ‘Case’). The allowNull: false constraint ensures a unit of measure is always recorded.
  + UnitPrice (DECIMAL(10, 2)): This field stores the price per unit of the product for this specific line item.
  + Notes (TEXT): An optional field for any specific notes related to this line item.
  + UOMID (INTEGER, foreign key referencing UnitOfMeasures): This foreign key links the line item to the specific unit of measure defined in the UnitOfMeasures model, providing a more robust and normalized way to manage units of measure.
* **Relationships:**
  + belongsTo OutboundOrder (foreignKey: OrderID): Each outbound order line item belongs to a specific outbound order.
  + belongsTo Product (foreignKey: ProductID): Each line item refers to a specific product.
  + belongsTo UnitOfMeasure (foreignKey: UOMID): Each line item is associated with a specific unit of measure.
* **Indexes:** There are no explicit indexes defined in this model schema. However, you might consider adding an index on OrderID to improve the performance of querying line items for a specific order.
* **Timestamps:** The timestamps: false configuration indicates that this model does not automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'outbound\_order\_line\_items' configuration specifies the database table name.
* **Example Use Cases:**
  + An outbound order might have a line item for 5 units of Product ‘B’.
  + Later, when a shipment is created, the QuantityShipped for this line item might be updated to 5.
  + The UnitOfMeasure could be ‘Pieces’, and the UnitPrice could be $10.00.

**4. OutboundOrder Model**

* **Purpose:** The OutboundOrder model represents an order placed by a customer to receive products from the warehouse. It contains header information about the order, such as the order number, date, customer details, and shipping information.
* **Attributes:**
  + OrderID (INTEGER, primaryKey, autoIncrement): The unique identifier for each outbound order.
  + OrderNumber (STRING, unique: true, allowNull: false): A unique, human-readable number assigned to each outbound order for easy identification and tracking. The unique: true constraint ensures that no two orders have the same order number, and allowNull: false mandates its presence.
  + OrderDate (DATE, allowNull: false): The date when the outbound order was placed by the customer.
  + CustomerID (INTEGER, foreign key referencing Customers): This foreign key links the outbound order to the customer who placed it. Note: The Customers model is not explicitly defined in your provided schemas, but it’s a logical association.
  + OrderStatus (STRING(50), defaultValue: 'Pending'): This field tracks the current status of the outbound order (e.g., ‘Pending’, ‘Processing’, ‘Shipped’, ‘Delivered’, ‘Cancelled’). It defaults to ‘Pending’ when a new order is created.
  + OrderCreatedByUserID (INTEGER, allowNull: false, foreign key referencing Users): This foreign key identifies the user within your system who created the outbound order (e.g., a sales representative).
  + ShippingAddressID (INTEGER, foreign key referencing Addresses): This foreign key links the order to the address where the products should be shipped.
  + BillingAddressID (INTEGER, foreign key referencing Addresses): This foreign key links the order to the address where the invoice should be sent.
  + ShippingMethod (STRING(100)): The method of shipping chosen for the order (e.g., ‘Standard’, ‘Express’).
  + ExpectedShipDate (DATEONLY): The date by which the order is expected to be shipped from the warehouse.
  + Notes (TEXT): An optional field for any general notes related to the outbound order.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the outbound order record was created.
* **Relationships:**
  + hasMany OutboundOrderLineItem (foreignKey: OrderID): Each outbound order can have multiple line items. The onDelete: 'CASCADE' on the OutboundOrderLineItem model ensures that deleting an order also deletes its line items.
  + belongsTo User (foreignKey: OrderCreatedByUserID): The order was created by a specific user.
  + belongsTo Address (foreignKey: ShippingAddressID, as: ‘ShippingAddress’): The order has a shipping address.
  + belongsTo Address (foreignKey: BillingAddressID, as: ‘BillingAddress’): The order has a billing address.
* **Indexes:**
  + unique index on ['OrderNumber']: Ensures that each outbound order has a unique order number.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'outbound\_orders' configuration specifies the database table name.
* **Example Use Cases:**
  + Recording a new order with order number ‘SO-2023-001’ placed on a specific date.
  + Tracking the status of the order as it moves through the fulfillment process.
  + Associating the order with the customer who placed it and the user who entered it into the system.
  + Storing the designated shipping and billing addresses.

**5. ProductCategory Model**

* **Purpose:** The ProductCategory model is used to categorize products within the WMS. This allows for better organization, filtering, and reporting of products based on their type or classification.
* **Attributes:**
  + CategoryID (INTEGER, primaryKey, autoIncrement): The unique identifier for each product category.
  + CategoryName (STRING(100), unique: true, allowNull: false): The name of the product category (e.g., ‘Electronics’, ‘Apparel’, ‘Food’). The unique: true constraint ensures that each category name is unique, and allowNull: false makes it a mandatory field.
  + Description (TEXT): An optional field providing a more detailed description of the product category.
* **Relationships:**
  + hasMany Product (foreignKey: CategoryID): Each product category can contain multiple products.
* **Indexes:**
  + unique index on ['CategoryName']: Ensures that each product category has a unique name.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'product\_categories' configuration specifies the database table name.
* **Example Use Cases:**
  + Creating categories like ‘Raw Materials’, ‘Finished Goods’, or ‘Spare Parts’.
  + Assigning products to their respective categories for inventory management and reporting.
  + Filtering product lists based on category.

**6. ProductLot Model**

* **Purpose:** The ProductLot model represents a specific batch or lot of a product. This is crucial for tracking products that are managed by lot numbers, often due to manufacturing dates, expiration dates, or quality control requirements.
* **Attributes:**
  + ProductLotID (INTEGER, primaryKey, autoIncrement): The unique identifier for each product lot.
  + ProductID (INTEGER, allowNull: false, foreign key referencing Products): This foreign key links the product lot to the specific product it belongs to. The allowNull: false constraint ensures that every lot is associated with a product.
  + LotNumber (STRING, allowNull: false): The unique lot number assigned to this batch of the product. The allowNull: false constraint indicates that a lot number must be provided.
  + ExpirationDate (DATEONLY): An optional field storing the expiration date of the product lot, if applicable.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the product lot record was created.
* **Relationships:**
  + belongsTo Product (foreignKey: ProductID): Each product lot belongs to a specific product.
  + hasMany Inventory (foreignKey: ProductLotID): Each product lot can have inventory in multiple warehouse locations.
  + hasMany StockTakeItem (foreignKey: ProductLotID): Each product lot can be included in multiple stock take items.
* **Indexes:**
  + unique index on ['ProductID', 'LotNumber'] (defined in uniqueKeys and indexes): This ensures that for a given product, the combination of ProductID and LotNumber is unique, preventing duplicate lot entries for the same product.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'product\_lots' configuration specifies the database table name.
* **Example Use Cases:**
  + Tracking different batches of a food product with varying expiration dates.
  + Managing inventory of electronic components based on their manufacturing lot.
  + Identifying specific recalls or quality issues related to a particular product lot.

**7. Product Model**

* **Purpose:** The Product model stores the fundamental information about each unique product managed within the WMS. This includes its code, name, description, physical attributes, and links to its category and base unit of measure.
* **Attributes:**
  + ProductID (INTEGER, primaryKey, autoIncrement): The unique identifier for each product.
  + ProductCode (STRING, unique: true, allowNull: false): A unique code or SKU assigned to the product for internal identification. The unique: true constraint ensures that each product has a distinct code, and allowNull: false makes it mandatory.
  + ProductName (STRING, allowNull: false): The descriptive name of the product. The allowNull: false constraint ensures that every product has a name.
  + Description (TEXT): An optional field providing a more detailed description of the product.
  + Dimensions (STRING): An optional field to store the dimensions of the product (e.g., length x width x height).
  + Weight (DECIMAL(10, 2)): An optional field to store the weight of the product, typically in a standard unit like kilograms or pounds. The DECIMAL(10, 2) type allows for storing decimal values with up to 10 digits in total and 2 digits after the decimal point.
  + CategoryID (INTEGER, foreign key referencing ProductCategory): This foreign key links the product to its specific product category.
  + UOMID (INTEGER, foreign key referencing UnitOfMeasure): This foreign key links the product to its base unit of measure (e.g., ‘Each’ for individual items).
* **Relationships:**
  + belongsTo ProductCategory (foreignKey: CategoryID): Each product belongs to a specific category.
  + belongsTo UnitOfMeasure (foreignKey: UOMID): Each product has a base unit of measure.
  + hasMany ProductLot (foreignKey: ProductID): Each product can have multiple lots.
  + hasMany PurchaseOrderLineItem (foreignKey: ProductID): A product can appear on multiple purchase order line items.
  + hasMany ReceiptLineItem (foreignKey: ProductID): A product can appear on multiple receipt line items.
  + hasMany OutboundOrderLineItem (foreignKey: ProductID): A product can appear on multiple outbound order line items.
  + hasMany ShipmentLineItem (foreignKey: ProductID): A product can be included in multiple shipment line items.
  + hasMany ReturnAuthorizationLineItem (foreignKey: ProductID): A product can be part of multiple return authorizations.
  + hasMany StockMovement (foreignKey: ProductID): Stock movements are tracked per product.
* **Indexes:**
  + unique index on ['ProductCode']: Ensures that each product has a unique product code.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName is implicitly ‘products’ based on the model name.
* **Example Use Cases:**
  + Creating a new product with code ‘ABC-123’, name ‘Widget’, and assigning it to the ‘Electronics’ category.
  + Storing the dimensions and weight of the widget for shipping calculations.
  + Referencing this product in purchase orders, receipts, and outbound orders.

**8. PurchaseOrderLineItem Model**

* **Purpose:** The PurchaseOrderLineItem model represents a single item within a purchase order. It details the specific product being ordered, the quantity, the unit price, the expected delivery date, and the unit of measure for that item.
* **Attributes:**
  + PurchaseOrderLineItemID (INTEGER, primaryKey, autoIncrement): The unique identifier for each line item within a purchase order.
  + PurchaseOrderID (INTEGER, allowNull: false, foreign key referencing PurchaseOrders, onDelete: 'CASCADE'): This foreign key links the line item to the specific purchase order it belongs to. The onDelete: 'CASCADE' setting ensures that if a purchase order is deleted, all its associated line items are also deleted.
  + ProductID (INTEGER, allowNull: false, foreign key referencing Products): This foreign key identifies the specific product being ordered in this line item.
  + QuantityOrdered (INTEGER, allowNull: false): The quantity of the product ordered in this line item.
  + UnitOfMeasure (STRING(50), allowNull: false): The unit of measure in which the product is being ordered (e.g., ‘Dozen’, ‘Pallet’).
  + UnitPrice (DECIMAL(10, 2)): The price per unit of the product as agreed upon in the purchase order.
  + ExpectedDeliveryDate (DATEONLY): The date when the ordered items are expected to be delivered by the supplier.
  + Notes (TEXT): An optional field for any specific notes related to this line item.
  + UOMID (INTEGER, foreign key referencing UnitOfMeasures): This foreign key provides a normalized link to the UnitOfMeasure model.
* **Relationships:**
  + belongsTo PurchaseOrder (foreignKey: PurchaseOrderID): Each purchase order line item belongs to a specific purchase order.
  + belongsTo Product (foreignKey: ProductID): Each line item refers to a specific product.
  + belongsTo UnitOfMeasure (foreignKey: UOMID): Each line item is associated with a specific unit of measure.
  + hasMany ReceiptLineItem (foreignKey: PurchaseOrderLineItemID): A purchase order line item can be fulfilled by one or more receipt line items.
* **Indexes:** There are no explicit indexes defined in this model schema. An index on PurchaseOrderID would be beneficial.
* **Timestamps:** The timestamps: false configuration indicates that this model does not automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'purchase\_order\_line\_items' configuration specifies the database table name.
* **Example Use Cases:**
  + A purchase order might include a line item for 100 units of Product ‘C’ at a price of $5.00 each, expected to be delivered by a specific date.
  + This line item can then be referenced when a receipt is created for the delivered goods.

**9. PurchaseOrder Model**

* **Purpose:** The PurchaseOrder model represents an order placed by your company to a supplier to procure goods for the warehouse. It contains header information such as the PO number, order date, supplier details, and expected delivery date.
* **Attributes:**
  + PurchaseOrderID (INTEGER, primaryKey, autoIncrement): The unique identifier for each purchase order.
  + PONumber (STRING, unique: true, allowNull: false): A unique, human-readable number assigned to each purchase order.
  + OrderDate (DATE, allowNull: false): The date when the purchase order was created and sent to the supplier.
  + SupplierID (INTEGER, allowNull: false, foreign key referencing Suppliers): This foreign key identifies the supplier from whom the goods are being ordered.
  + OrderCreatedByUserID (INTEGER, allowNull: false, foreign key referencing Users): This foreign key identifies the user who created the purchase order.
  + OrderStatus (STRING(50), defaultValue: 'Pending'): The current status of the purchase order (e.g., ‘Pending’, ‘Sent’, ‘Partially Received’, ‘Received’, ‘Cancelled’).
  + TotalAmount (DECIMAL(15, 2)): The total monetary value of the purchase order.
  + ExpectedDeliveryDate (DATEONLY): The date by which the ordered goods are expected to be delivered by the supplier.
  + ShippingAddressID (INTEGER, foreign key referencing Addresses): The address where the ordered goods should be shipped.
  + BillingAddressID (INTEGER, foreign key referencing Addresses): The address where the supplier should send the invoice.
  + Notes (TEXT): Any general notes or instructions related to the purchase order.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the purchase order record was created.
* **Relationships:**
  + hasMany PurchaseOrderLineItem (foreignKey: PurchaseOrderID): Each purchase order can have multiple line items. The onDelete: 'CASCADE' on PurchaseOrderLineItem ensures that deleting a PO also deletes its line items.
  + belongsTo Supplier (foreignKey: SupplierID): The purchase order is for a specific supplier.
  + belongsTo User (foreignKey: OrderCreatedByUserID): The purchase order was created by a specific user.
  + belongsTo Address (foreignKey: ShippingAddressID, as: ‘ShippingAddress’): The purchase order has a shipping address.
  + belongsTo Address (foreignKey: BillingAddressID, as: ‘BillingAddress’): The purchase order has a billing address.
  + hasMany Receipt (foreignKey: PurchaseOrderID): One purchase order can be fulfilled by multiple receipts (e.g., partial deliveries).
* **Indexes:**
  + unique index on ['PONumber']: Ensures that each purchase order has a unique PO number.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName is implicitly ‘purchase\_orders’ based on the model name.
* **Example Use Cases:**
  + Creating a new purchase order with PO number ‘PO-2023-005’ for a specific supplier.
  + Tracking the status of the PO as it’s processed and goods are received.
  + Associating the PO with the user who created it and the designated shipping and billing addresses.

**10. ReceiptLineItem Model**

* **Purpose:** The ReceiptLineItem model represents a single item received as part of a receipt. It details the product received, the expected and received quantities, the lot number, expiration date, and other relevant information captured upon receiving goods.
* **Attributes:**
  + ReceiptLineItemID (INTEGER, primaryKey, autoIncrement): The unique identifier for each line item within a receipt.
  + ReceiptID (INTEGER, allowNull: false, foreign key referencing Receipts, onDelete: 'CASCADE'): This foreign key links the line item to the specific receipt it belongs to. The onDelete: 'CASCADE' ensures that deleting a receipt also deletes its line items.
  + ProductID (INTEGER, foreign key referencing Products): This foreign key identifies the specific product received in this line item.
  + ExpectedQuantity (INTEGER, allowNull: false): The quantity of the product that was expected to be received according to the related purchase order or advance shipment notice.
  + ReceivedQuantity (INTEGER, allowNull: false): The actual quantity of the product that was received.
  + UnitOfMeasure (STRING(50), allowNull: false): The unit of measure in which the product was received.
  + LotNumber (STRING): The lot number of the received product, if applicable.
  + ExpirationDate (DATEONLY): The expiration date of the received product lot, if applicable.
  + SerialNumber (STRING): The serial number of the received product, if applicable.
  + ConditionOnReceipt (TEXT): A description of the condition of the product upon receipt.
  + CountryOfOrigin (STRING(100)): The country where the product was originally manufactured.
  + UOMID (INTEGER, foreign key referencing UnitOfMeasures): A normalized link to the UnitOfMeasures model.
  + PurchaseOrderLineItemID (INTEGER, foreign key referencing PurchaseOrderLineItems): An optional foreign key linking this receipt line item back to the specific line item on the purchase order that it fulfills.
* **Relationships:**
  + belongsTo Receipt (foreignKey: ReceiptID): Each receipt line item belongs to a specific receipt.
  + belongsTo Product (foreignKey: ProductID): Each line item refers to a specific product.
  + belongsTo UnitOfMeasure (foreignKey: UOMID): Each line item is associated with a specific unit of measure.
  + belongsTo PurchaseOrderLineItem (foreignKey: PurchaseOrderLineItemID): This line item can be linked to a specific purchase order line item.
* **Indexes:** An index on ReceiptID would be beneficial.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'receipt\_line\_item' configuration specifies the database table name.
* **Example Use Cases:**
  + Recording that for a specific receipt, 50 units of Product ‘D’ were received, even though 55 were expected according to the purchase order.
  + Capturing the lot number and expiration date of a perishable item upon arrival.
  + Noting any damage or specific conditions observed during the receiving process.

**11. Receipt Model**

* **Purpose:** The Receipt model represents a record of goods received into the warehouse. It contains header information about the delivery, such as the receipt date, carrier details, purchase order reference, and the users involved in the receiving process.
* **Attributes:**
  + ReceiptID (INTEGER, primaryKey, autoIncrement): The unique identifier for each receipt record.
  + ReceiptDate (DATE, allowNull: false): The date when the goods were received into the warehouse.
  + ExpectedDeliveryDate (DATEONLY): The date when the delivery was originally expected.
  + CarrierName (STRING): The name of the carrier that delivered the goods.
  + TrackingNumber (STRING): The tracking number associated with the delivery.
  + PONumber (STRING): The purchase order number related to this receipt.
  + NumberOfPallets (INTEGER): The number of pallets received.
  + SealNumbers (STRING): Any seal numbers present on the delivery vehicle or containers.
  + TemperatureOnArrival (DECIMAL(5, 2)): The temperature of the goods upon arrival, if relevant (e.g., for temperature-sensitive items).
  + ConditionOnArrival (TEXT): A general description of the condition of the delivery upon arrival.
  + DriverName (STRING): The name of the driver who delivered the goods.
  + UnloadingStartTime (DATE): The time when the unloading of the delivery began.
  + UnloadingEndTime (DATE): The time when the unloading of the delivery was completed.
  + ReceivingBayLocation (STRING(50)): The specific location within the receiving area where the goods were initially placed.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the receipt record was created.
  + Notes (TEXT): Any general notes or comments related to the receipt.
  + InspectionStatus (STRING(50)): The status of the inspection process for the received goods (e.g., ‘Pending’, ‘Completed’, ‘Passed’, ‘Failed’).
  + InspectionDateTime (DATE): The date and time when the inspection was performed.
  + SampleSizeInspected (INTEGER): The number of units or items inspected.
  + SupplierID (INTEGER, foreign key referencing Suppliers): The supplier from whom the goods were received.
  + ReceiverUserID (INTEGER, foreign key referencing Users, as: ‘Receiver’): The user who received the goods.
  + InspectorUserID (INTEGER, foreign key referencing Users, as: ‘Inspector’): The user who performed the inspection, if applicable.
  + PurchaseOrderID (INTEGER, foreign key referencing PurchaseOrders): The purchase order that this receipt is fulfilling.
* **Relationships:**
  + hasMany ReceiptLineItem (foreignKey: ReceiptID): Each receipt can have multiple line items detailing the received products. The onDelete: 'CASCADE' on ReceiptLineItem ensures that deleting a receipt also deletes its line items.
  + belongsTo Supplier (foreignKey: SupplierID): The receipt is for goods received from a specific supplier.
  + belongsTo User (foreignKey: ReceiverUserID, as: ‘Receiver’): The receipt is associated with the user who performed the receiving.
  + belongsTo User (foreignKey: InspectorUserID, as: ‘Inspector’): The receipt can be associated with a user who performed the inspection.
  + belongsTo PurchaseOrder (foreignKey: PurchaseOrderID): The receipt is often related to a specific purchase order.
* **Indexes:** An index on PurchaseOrderID and SupplierID would be useful for querying receipts.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'receipt' configuration specifies the database table name.
* **Example Use Cases:**
  + Recording the details of a delivery received on a specific date from a particular carrier.
  + Referencing the associated purchase order and noting any discrepancies or issues with the delivery.
  + Tracking the time taken to unload the goods and the initial receiving bay location.

**12. ReturnAuthorizationLineItem Model**

* **Purpose:** The ReturnAuthorizationLineItem model represents a single item that is being authorized for return by a customer. It links back to the original outbound order line item and specifies the quantity being returned and the reason.
* **Attributes:**
  + RMALineItemID (INTEGER, primaryKey, autoIncrement): The unique identifier for each line item within a return authorization.
  + RMAID (INTEGER, allowNull: false, foreign key referencing ReturnAuthorizations, onDelete: 'CASCADE'): This foreign key links the line item to the specific return authorization it belongs to. The onDelete: 'CASCADE' ensures that if a return authorization is deleted, all its associated line items are also deleted.
  + OrderItemID (INTEGER, foreign key referencing OutboundOrderLineItems): This foreign key links the return authorization line item back to the original line item in the outbound order that the customer is returning.
  + ProductID (INTEGER, allowNull: false, foreign key referencing Products): This foreign key identifies the specific product being returned.
  + QuantityReturned (INTEGER, allowNull: false): The quantity of the product that the customer is authorized to return.
  + ConditionOnReturn (TEXT): A description of the condition of the product as reported by the customer or upon initial inspection.
  + Notes (TEXT): Any additional notes related to this specific return authorization line item.
* **Relationships:**
  + belongsTo ReturnAuthorization (foreignKey: RMAID): Each return authorization line item belongs to a specific return authorization.
  + belongsTo OutboundOrderLineItem (foreignKey: OrderItemID): Each line item refers back to the original outbound order line item.
  + belongsTo Product (foreignKey: ProductID): Each line item specifies the product being returned.
* **Indexes:** An index on RMAID and OrderItemID would be beneficial.
* **Timestamps:** The timestamps: false configuration indicates that this model does not automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'return\_authorization\_line\_items' configuration specifies the database table name.
* **Example Use Cases:**
  + A customer might request a return for 2 units of a product from a specific order due to damage.
  + This model would store that request, linking it back to the original order and product.

**13. ReturnAuthorization Model**

* **Purpose:** The ReturnAuthorization model (often abbreviated as RMA - Return Merchandise Authorization) represents an authorization granted to a customer to return goods. It contains header information about the return request, the associated order, and the reason for the return.
* **Attributes:**
  + RMAID (INTEGER, primaryKey, autoIncrement): The unique identifier for each return authorization.
  + RMANumber (STRING, unique: true, allowNull: false): A unique number assigned to each return authorization for tracking purposes.
  + OrderID (INTEGER, foreign key referencing OutboundOrders): This foreign key links the return authorization to the original outbound order from which the return originates.
  + ShipmentID (INTEGER, foreign key referencing Shipments): This foreign key can optionally link the return authorization to a specific shipment associated with the original order.
  + ReturnDate (DATE, defaultValue: DataTypes.NOW): The date when the return authorization was created.
  + ReasonForReturn (STRING(255)): A brief description of the reason why the customer is requesting a return.
  + ReturnStatus (STRING(50), defaultValue: 'Pending'): The current status of the return authorization (e.g., ‘Pending’, ‘Approved’, ‘Rejected’, ‘Received’, ‘Completed’).
  + RequestedByUserID (INTEGER, foreign key referencing Users): The user within your system who initiated the return authorization (e.g., a customer service agent).
  + ApprovedByUserID (INTEGER, foreign key referencing Users): The user who approved the return authorization, if approval is required.
  + Notes (TEXT): Any general notes or comments related to the return authorization.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the return authorization record was created.
* **Relationships:**
  + hasMany ReturnAuthorizationLineItem (foreignKey: RMAID): Each return authorization can have multiple line items for different products or quantities. The onDelete: 'CASCADE' on ReturnAuthorizationLineItem ensures that deleting an RMA also deletes its line items.
  + belongsTo OutboundOrder (foreignKey: OrderID): The RMA is related to a specific outbound order.
  + belongsTo Shipment (foreignKey: ShipmentID): The RMA can be optionally linked to a shipment.
  + belongsTo User (foreignKey: RequestedByUserID, as: ‘RequestedBy’): The RMA was requested by a specific user.
  + belongsTo User (foreignKey: ApprovedByUserID, as: ‘ApprovedBy’): The RMA can be approved by a specific user.
* **Indexes:**
  + unique index on ['RMANumber']: Ensures that each return authorization has a unique number.
* **Timestamps:** The timestamps: false configuration and freezeTableName: true suggest a more explicit control over table and timestamp management for this model.
* **Table Name:** The tableName: 'return\_authorizations' is implied by the model name, though freezeTableName: true might prevent Sequelize from pluralizing it.
* **Example Use Cases:**
  + Generating a new RMA number for a customer’s return request.
  + Linking the RMA to the original order and specifying the reason for the return.
  + Tracking the approval process and the eventual receipt of the returned goods.

**14. Role Model**

* **Purpose:** The Role model defines the different roles that users can have within the WMS. This is fundamental for implementing role-based access control (RBAC), where different roles have different permissions and access levels within the system.
* **Attributes:**
  + RoleID (INTEGER, primaryKey, autoIncrement): The unique identifier for each role.
  + RoleName (STRING(100), unique: true, allowNull: false): The name of the role (e.g., ‘Warehouse Manager’, ‘Inventory Clerk’, ‘Shipping Operator’). The unique: true constraint ensures that each role name is unique.
  + Description (TEXT): An optional field providing a description of the responsibilities and permissions associated with this role.
* **Relationships:**
  + belongsToMany User (through UserRole): A user can have multiple roles, and a role can be assigned to multiple users. This many-to-many relationship is mediated by the UserRole model.
* **Indexes:**
  + unique index on ['RoleName']: Ensures that each role has a unique name.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'roles' configuration specifies the database table name.
* **Example Use Cases:**
  + Defining roles with specific permissions, such as the ability to create outbound orders, manage inventory, or process shipments.
  + Assigning these roles to users based on their job responsibilities.

**15. ShipmentLineItem Model**

* **Purpose:** The ShipmentLineItem model represents a single item included in a shipment. It details the specific product, the quantity shipped, and links back to the original outbound order line item.
* **Attributes:**
  + ShipmentLineItemID (INTEGER, primaryKey, autoIncrement): The unique identifier for each line item within a shipment.
  + ShipmentID (INTEGER, allowNull: false, foreign key referencing Shipments, onDelete: 'CASCADE'): This foreign key links the line item to the specific shipment it belongs to. The onDelete: 'CASCADE' ensures that if a shipment is deleted, all its associated line items are also deleted.
  + OrderItemID (INTEGER, foreign key referencing OutboundOrderLineItems): This foreign key links the shipment line item back to the original line item in the outbound order that is being fulfilled by this shipment.
  + ProductID (INTEGER, allowNull: false, foreign key referencing Products): This foreign key identifies the specific product being shipped.
  + QuantityShipped (INTEGER, allowNull: false): The quantity of the product included in this shipment line item.
  + LotNumber (STRING): The lot number of the shipped product, if applicable.
  + SerialNumber (STRING): The serial number of the shipped product, if applicable.
* **Relationships:**
  + belongsTo Shipment (foreignKey: ShipmentID): Each shipment line item belongs to a specific shipment.
  + belongsTo OutboundOrderLineItem (foreignKey: OrderItemID): Each line item fulfills a part of an outbound order line item.
  + belongsTo Product (foreignKey: ProductID): Each line item specifies the product being shipped.
* **Indexes:** An index on ShipmentID and OrderItemID would be beneficial.
* **Timestamps:** The timestamps: false configuration indicates that this model does not automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'shipment\_line\_items' configuration specifies the database table name.
* **Example Use Cases:**
  + A shipment might contain 3 units of Product ‘E’ from a specific outbound order.
  + This model would record that detail, linking it to the shipment and the original order.

**16. Shipment Model**

* **Purpose:** The Shipment model represents a physical delivery of goods to a customer. It contains header information about the shipment, such as the carrier, tracking number, and the associated outbound order.
* **Attributes:**
  + ShipmentID (INTEGER, primaryKey, autoIncrement): The unique identifier for each shipment.
  + OrderID (INTEGER, allowNull: false, foreign key referencing OutboundOrders): This foreign key links the shipment to the specific outbound order it is fulfilling.
  + ShipmentNumber (STRING, unique: true, allowNull: false): A unique number assigned to each shipment for tracking purposes.
  + ShipmentDate (DATE, defaultValue: DataTypes.NOW): The date when the shipment was created or dispatched.
  + CarrierName (STRING): The name of the shipping carrier (e.g., ‘UPS’, ‘FedEx’).
  + TrackingNumber (STRING): The tracking number provided by the carrier.
  + ShippingCost (DECIMAL(10, 2)): The cost of the shipment.
  + ShippedByUserID (INTEGER, foreign key referencing Users): The user who processed and initiated the shipment.
  + DepartureTime (DATE): The time when the shipment departed from the warehouse.
  + EstimatedArrivalTime (DATE): The estimated time of arrival at the customer’s location.
  + DeliveryConfirmationNumber (STRING): The delivery confirmation number, if available.
  + DeliveryDateTime (DATE): The date and time when the shipment was delivered.
  + Notes (TEXT): Any additional notes related to the shipment.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the shipment record was created.
* **Relationships:**
  + belongsTo OutboundOrder (foreignKey: OrderID): Each shipment fulfills a specific outbound order.
  + hasMany ShipmentLineItem (foreignKey: ShipmentID): Each shipment can contain multiple line items detailing the products being shipped. The onDelete: 'CASCADE' on ShipmentLineItem ensures that deleting a shipment also deletes its line items.
  + belongsTo User (foreignKey: ShippedByUserID): The shipment was processed by a specific user.
  + hasMany ReturnAuthorization (foreignKey: ShipmentID): One shipment might be associated with multiple return authorizations (though this relationship might need further scrutiny based on your exact workflow).
* **Indexes:**
  + unique index on ['ShipmentNumber']: Ensures that each shipment has a unique number.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'shipments' configuration specifies the database table name.
* **Example Use Cases:**
  + Creating a new shipment record for an outbound order, including the carrier and tracking number.
  + Updating the shipment status with departure and delivery information.
  + Calculating the shipping cost for reporting purposes.

**17. StockMovement Model**

* **Purpose:** The StockMovement model tracks all movements of inventory within the warehouse. This is crucial for maintaining accurate inventory records, auditing stock levels, and understanding the flow of goods.
* **Attributes:**
  + StockMovementID (INTEGER, primaryKey, autoIncrement): The unique identifier for each stock movement record.
  + MovementType (ENUM('Inbound', 'Outbound', 'Transfer', 'Adjustment'), allowNull: false): The type of stock movement, indicating whether it was an incoming shipment, an outgoing shipment, a transfer between locations, or an inventory adjustment. The ENUM type restricts the value to one of these predefined options.
  + ProductID (INTEGER, allowNull: false, foreign key referencing Products): The specific product that was moved.
  + FromLocationID (INTEGER, foreign key referencing WarehouseLocations): The location from which the stock was moved. This can be null for inbound movements.
  + ToLocationID (INTEGER, foreign key referencing WarehouseLocations): The location to which the stock was moved. This can be null for outbound movements.
  + Quantity (INTEGER, allowNull: false): The quantity of the product that was moved.
  + MovementDate (DATE, defaultValue: DataTypes.NOW): The date and time when the stock movement occurred.
  + ReferenceID (INTEGER): An optional ID of the related record (e.g., ReceiptID, ShipmentID, OrderID).
  + ReferenceType (STRING(50)): An optional type of the related record (e.g., ‘Receipt’, ‘Shipment’, ‘Order’, ‘StockTake’).
  + Reason (STRING(255)): An optional reason for the stock movement, especially for adjustments or transfers.
  + UserID (INTEGER, foreign key referencing Users): The user who initiated or recorded the stock movement.
  + Notes (TEXT): Any additional notes related to the stock movement.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the stock movement record was created.
* **Relationships:**
  + belongsTo Product (foreignKey: ProductID): Each stock movement involves a specific product.
  + belongsTo WarehouseLocation (foreignKey: FromLocationID, as: ‘FromLocation’): The location from which the stock moved.
  + belongsTo WarehouseLocation (foreignKey: ToLocationID, as: ‘ToLocation’): The location to which the stock moved.
  + belongsTo User (foreignKey: UserID): The stock movement was performed by a specific user.
  + belongsTo StockTakeItem (foreignKey: AdjustmentStockMovementID): Stock movements of type ‘Adjustment’ can be linked to a specific stock take item that triggered the adjustment.
* **Indexes:** An index on ProductID, FromLocationID, ToLocationID, MovementType, and MovementDate would be useful for querying stock movements.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'stock\_movements' configuration specifies the database table name.
* **Example Use Cases:**
  + Recording an inbound receipt of 100 units of a product.
  + Tracking the shipment of 50 units of a product to a customer.
  + Logging the transfer of 10 units of a product from a staging area to a storage location.
  + Recording an inventory adjustment where 5 units were written off due to damage.

**18. StockTakeItem Model**

* **Purpose:** The StockTakeItem model represents a single item counted during a stock take process. It links to the overall StockTake and records the expected and counted quantities of a specific product lot at a particular location.
* **Attributes:**
  + StockTakeItemID (INTEGER, primaryKey, autoIncrement): The unique identifier for each item counted during a stock take.
  + StockTakeID (INTEGER, allowNull: false, foreign key referencing StockTakes, onDelete: 'CASCADE'): This foreign key links the counted item to the specific stock take event it belongs to. The onDelete: 'CASCADE' ensures that deleting a stock take also deletes its items.
  + InventoryID (INTEGER, foreign key referencing Inventories): An optional foreign key linking this stock take item to an existing inventory record.
  + ProductLotID (INTEGER, allowNull: false, foreign key referencing ProductLots): The specific product lot that was counted.
  + LocationID (INTEGER, allowNull: false, foreign key referencing WarehouseLocations): The location where the product lot was counted.
  + ExpectedQuantity (INTEGER): The quantity of the product lot that was expected to be at this location according to the system records.
  + CountedQuantity (INTEGER, allowNull: false, defaultValue: 0): The actual quantity of the product lot that was physically counted.
  + CountedByUserID (INTEGER, foreign key referencing Users): The user who performed the count for this item.
  + CountedDate (DATE, defaultValue: DataTypes.NOW): The date and time when the count was performed.
  + Discrepancy (INTEGER, calculated property): This virtual attribute calculates the difference between the ExpectedQuantity and the CountedQuantity.
  + ReasonForDiscrepancy (STRING(255)): An optional reason provided for any discrepancy found.
  + AdjustmentStockMovementID (INTEGER, foreign key referencing StockMovements): An optional foreign key linking to a StockMovement record if an inventory adjustment was made based on this stock take item.
  + Notes (TEXT): Any additional notes related to this specific stock take item.
* **Relationships:**
  + belongsTo StockTake (foreignKey: StockTakeID): Each stock take item belongs to a specific stock take event.
  + belongsTo Inventory (foreignKey: InventoryID): The stock take item can be linked to an existing inventory record.
  + belongsTo ProductLot (foreignKey: ProductLotID): The item counted is a specific product lot.
  + belongsTo WarehouseLocation (foreignKey: LocationID): The item was counted at a specific location.
  + belongsTo User (foreignKey: CountedByUserID): The item was counted by a specific user.
  + belongsTo StockMovement (foreignKey: AdjustmentStockMovementID): The stock take item can be linked to an adjustment stock movement.
* **Indexes:** An index on StockTakeID, ProductLotID, and LocationID would be beneficial.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'stock\_take\_items' configuration specifies the database table name.
* **Example Use Cases:**
  + Recording that during a stock take, 98 units of a product lot were counted in a location, while the system expected 100.
  + Noting the reason for the discrepancy and linking it to an inventory adjustment if necessary.

**19. StockTake Model**

* **Purpose:** The StockTake model represents a specific inventory stock take event. It contains header information about the stock take, such as the start and end dates, the user who initiated it, and its current status.
* **Attributes:**
  + StockTakeID (INTEGER, primaryKey, autoIncrement): The unique identifier for each stock take event.
  + StockTakeNumber (STRING(50), unique: true, allowNull: false): A unique number assigned to each stock take for identification and tracking.
  + StartDate (DATE, allowNull: false): The date when the stock take began.
  + EndDate (DATE): The date when the stock take was completed.
  + InitiatedByUserID (INTEGER, allowNull: false, foreign key referencing Users): The user who initiated the stock take process.
  + Status (ENUM('Planning', 'InProgress', 'Completed', 'Verified', 'Cancelled'), defaultValue: 'Planning', allowNull: false): The current status of the stock take event.
  + Notes (TEXT): Any general notes or instructions related to the stock take.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the stock take record was created.
* **Relationships:**
  + hasMany StockTakeItem (foreignKey: StockTakeID): Each stock take event can have multiple stock take items representing the counted products. The onDelete: 'CASCADE' on StockTakeItem ensures that deleting a stock take also deletes its items.
  + belongsTo User (foreignKey: InitiatedByUserID): The stock take was initiated by a specific user.
* **Indexes:**
  + unique index on ['StockTakeNumber']: Ensures that each stock take has a unique number.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'stock\_takes' configuration specifies the database table name.
* **Example Use Cases:**
  + Creating a new stock take event with a specific number and start date.
  + Tracking the progress of the stock take by updating its status.
  + Recording the user who initiated the stock take.

**20. Supplier Model**

* **Purpose:** The Supplier model stores information about the companies or entities that supply goods to your warehouse.
* **Attributes:**
  + SupplierID (INTEGER, primaryKey, autoIncrement): The unique identifier for each supplier.
  + SupplierName (STRING, allowNull: false): The name of the supplier.
  + AddressID (INTEGER, foreign key referencing Addresses): A foreign key linking to the primary address of the supplier.
* **Relationships:**
  + belongsTo Address (foreignKey: AddressID): Each supplier has a primary address.
  + hasMany PurchaseOrder (foreignKey: SupplierID): You can place multiple purchase orders with one supplier.
  + hasMany Receipt (foreignKey: SupplierID): You can receive multiple shipments (receipts) from one supplier.
* **Indexes:** There are no explicit indexes defined, but you might consider an index on SupplierName.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'supplier' configuration specifies the database table name.
* **Example Use Cases:**
  + Recording the name and address of a new vendor.
  + Linking purchase orders and receipts to the correct supplier.

**21. UnitOfMeasure Model**

* **Purpose:** The UnitOfMeasure model defines the different units of measure used for products within the WMS (e.g., ‘Each’, ‘Box’, ‘Case’, ‘Pallet’). This ensures consistency and allows for conversions if needed.
* **Attributes:**
  + UOMID (INTEGER, primaryKey, autoIncrement): The unique identifier for each unit of measure.
  + UOMCode (STRING(20), unique: true, allowNull: false): A short, unique code for the unit of measure (e.g., ‘EA’, ‘BX’, ‘CS’, ‘PL’).
  + UOMName (STRING(100), allowNull: false): The full name of the unit of measure (e.g., ‘Each’, ‘Box’, ‘Case’, ‘Pallet’).
  + Description (TEXT): An optional field providing a description of the unit of measure.
* **Relationships:**
  + hasMany Product (foreignKey: UOMID): Each product has a base unit of measure.
  + hasMany PurchaseOrderLineItem (foreignKey: UOMID): Items on a purchase order are ordered in a specific unit of measure.
  + hasMany ReceiptLineItem (foreignKey: UOMID): Items received are recorded with a unit of measure.
  + hasMany OutboundOrderLineItem (foreignKey: UOMID): Items on an outbound order are ordered in a specific unit of measure.
* **Indexes:**
  + unique index on ['UOMCode']: Ensures that each unit of measure has a unique code.
* **Timestamps:** There are no explicit timestamps defined for this model in the provided schema.
* **Table Name:** The tableName: 'unit\_of\_measure' configuration specifies the database table name.
* **Example Use Cases:**
  + Defining standard units of measure used in your operations.
  + Ensuring that all product quantities are consistently recorded with the correct unit.

**22. UserRole Model**

* **Purpose:** The UserRole model is a junction table that establishes the many-to-many relationship between the User and Role models. It indicates which roles are assigned to which users.
* **Attributes:**
  + UserRoleID (INTEGER, primaryKey, autoIncrement): The unique identifier for each user-role assignment.
  + UserID (INTEGER, allowNull: false, foreign key referencing Users): The ID of the user.
  + RoleID (INTEGER, allowNull: false, foreign key referencing Roles): The ID of the role assigned to the user.
* **Relationships:**
  + belongsTo User (foreignKey: UserID): Each user-role assignment belongs to a specific user.
  + belongsTo Role (foreignKey: RoleID): Each user-role assignment is for a specific role.
* **Indexes:**
  + unique index on ['UserID', 'RoleID'] (defined in uniqueKeys and indexes): Ensures that a user can only be assigned a specific role once.
* **Timestamps:** The timestamps: false configuration indicates that this model does not automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'user\_roles' configuration specifies the database table name.
* **Example Use Cases:**
  + Assigning the ‘Warehouse Manager’ role to a specific user.
  + Allowing a user to have multiple roles, such as both ‘Inventory Clerk’ and ‘Shipping Operator’.

**23. User Model**

* **Purpose:** The User model stores information about the individuals who interact with the WMS.
* **Attributes:**
  + UserID (INTEGER, primaryKey, autoIncrement): The unique identifier for each user.
  + Username (STRING, unique: true, allowNull: false): The unique username used by the user to log in to the system.
* **Relationships:**
  + hasMany OutboundOrder (foreignKey: OrderCreatedByUserID): Tracks which user created an outbound order.
  + hasMany PurchaseOrder (foreignKey: OrderCreatedByUserID): Tracks which user created a purchase order.
  + hasMany Receipt (foreignKey: ReceiverUserID, as: ‘Receiver’): Tracks which user received a shipment.
  + hasMany Receipt (foreignKey: InspectorUserID, as: ‘Inspector’): Tracks which user inspected a shipment.
  + hasMany ReturnAuthorization (foreignKey: RequestedByUserID): Tracks which user requested a return authorization.
  + hasMany ReturnAuthorization (foreignKey: ApprovedByUserID): Tracks which user approved a return authorization.
  + hasMany Shipment (foreignKey: ShippedByUserID): Tracks which user processed a shipment.
  + hasMany StockMovement (foreignKey: UserID): Tracks which user initiated a stock movement.
  + hasMany StockTake (foreignKey: InitiatedByUserID): Tracks which user initiated a stock take.
  + hasMany StockTakeItem (foreignKey: CountedByUserID): Tracks which user counted an item during a stock take.
  + belongsToMany Role (through UserRole): Establishes the many-to-many relationship with the Role model.
* **Indexes:**
  + unique index on ['Username']: Ensures that each user has a unique username.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'user' configuration specifies the database table name.
* **Example Use Cases:**
  + Creating new user accounts for warehouse staff.
  + Tracking the actions performed by different users within the system.

**24. WarehouseLocation Model**

* **Purpose:** The WarehouseLocation model defines the different physical locations within the warehouse where inventory can be stored. It allows for a hierarchical structure of locations (e.g., Warehouse -> Zone -> Aisle -> Rack -> Shelf -> Bin).
* **Attributes:**
  + LocationID (INTEGER, primaryKey, autoIncrement): The unique identifier for each warehouse location.
  + LocationCode (STRING(50), unique: true, allowNull: false): A unique code for the location (e.g., ‘WH-A’, ‘ZONE-1’, ‘A-05’, ‘R-3-2’).
  + LocationName (STRING(100)): A descriptive name for the location.
  + LocationType (ENUM('Warehouse', 'Zone', 'Aisle', 'Rack', 'Shelf', 'Bin', 'Chamber', 'Other'), allowNull: false): The type of the location, defining its level in the warehouse hierarchy.
  + ParentLocationID (INTEGER, foreign key referencing WarehouseLocations, allowNull: true): A foreign key linking this location to its parent location in the hierarchy. Top-level locations (e.g., Warehouses) will have a null ParentLocationID.
  + Capacity (INTEGER): An optional field to store the capacity of the location (e.g., maximum number of items or volume).
  + Notes (TEXT): Any additional notes related to the location.
  + SystemTimestamp (DATE, defaultValue: DataTypes.NOW): The timestamp when the location record was created.
* **Relationships:**
  + belongsTo WarehouseLocation (foreignKey: ParentLocationID, as: ‘Parent’): Defines the parent-child relationship in the location hierarchy.
  + hasMany WarehouseLocation (foreignKey: ParentLocationID, as: ‘Children’): Allows a location to have multiple child locations.
  + hasMany Inventory (foreignKey: LocationID): Tracks the inventory stored at this location.
  + hasMany StockMovement (foreignKey: FromLocationID, as: ‘DepartingMovements’): Tracks stock moved from this location.
  + hasMany StockMovement (foreignKey: ToLocationID, as: ‘ArrivingMovements’): Tracks stock moved to this location.
  + hasMany StockTakeItem (foreignKey: LocationID): Tracks items counted at this location during a stock take.
* **Indexes:**
  + unique index on ['LocationCode']: Ensures that each warehouse location has a unique code.
* **Timestamps:** The timestamps: true configuration indicates that this model will automatically manage createdAt and updatedAt timestamps.
* **Table Name:** The tableName: 'warehouse\_locations' configuration specifies the database table name.
* **Example Use Cases:**
  + Defining the physical layout of your warehouse, including different zones, aisles, racks, and bins.
  + Assigning inventory to specific locations within the warehouse.
  + Tracking stock movements between different locations.

This detailed explanation should provide a solid understanding of each of your database models and their role in your Warehouse Management System. Remember that the relationships defined between these models are crucial for the system’s functionality, allowing you to track the flow of goods, manage inventory levels, and process orders effectively.