

Trucking Logistics Logical Model

Grumpy Old IT Guy Publishers and **Angelo R. Bobak** are independent educational content providers and are not affiliated with, endorsed by, sponsored by, or associated with Microsoft Corporation.

All references to Microsoft technologies are used strictly for educational, instructional, commentary, and demonstration purposes under the principles of nominative fair use.

Any screenshots, product references, feature descriptions, or code examples referencing Microsoft technologies are intended solely to help users learn database design, SQL development, and related technical concepts.

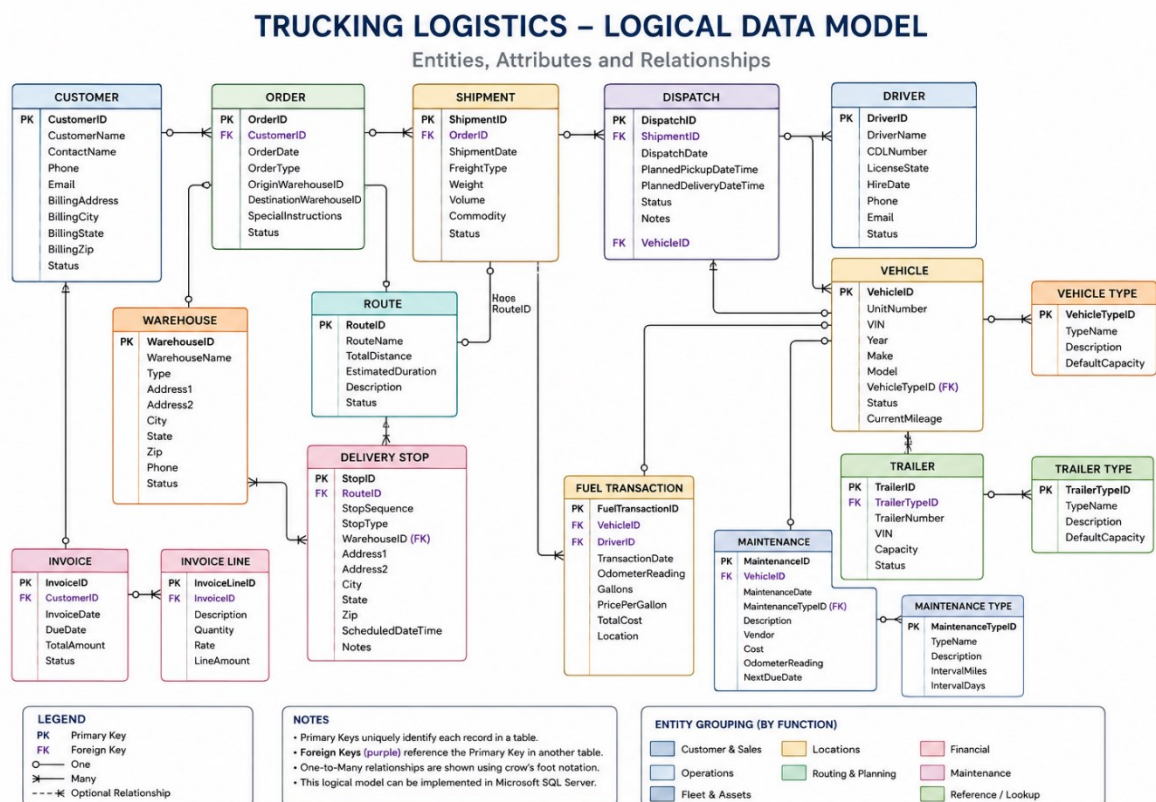


Figure 1 – Trucking Logistics Logical Data Model

The logical data model shown in **Figure 2 – Trucking Logistics Logical Data Model** expands upon the earlier conceptual model by introducing detailed entities, business attributes, primary keys, foreign keys, and operational relationships required to support a transportation management platform.

While the conceptual model focuses on major business functions, the logical model goes one level deeper by identifying the data elements needed to build a functioning operational database.

This design remains platform-independent from a physical implementation perspective, but it introduces the structural detail necessary to later build the Microsoft SQL Server database.

This model was intentionally designed to simulate a mid-sized trucking organization responsible for regional and long-haul freight operations, including customer relationships, warehouse operations, fleet assets, route planning, maintenance, billing, and operational cost tracking.

1. Customer Management Layer

CUSTOMER

The **Customer** table stores organizations or individuals that request transportation services.

Key attributes include:

- CustomerID (Primary Key)
- CustomerName
- ContactName
- Phone
- Email
- BillingAddress
- BillingCity
- BillingState
- BillingZip
- Status

This entity supports both recurring commercial customers and one-time shipping customers.

Business Rule:

One customer may create many orders over time.

2. Order Processing Layer

ORDER

The **Order** table captures customer freight requests before physical transportation begins.

Important attributes include:

- OrderID (Primary Key)
- CustomerID (Foreign Key)
- OrderDate
- OrderType
- OriginWarehouseID
- DestinationWarehouseID
- SpecialInstructions
- Status

This table acts as the intake mechanism for shipment processing.

Business Rule:

One customer can create many orders.

3. Shipment Management Layer

SHIPMENT

The **Shipment** table represents actual freight movements generated from customer orders.

Key attributes include:

- ShipmentID (Primary Key)
- OrderID (Foreign Key)
- ShipmentDate
- FreightType
- Weight
- Volume
- Commodity
- Status

This entity becomes the operational center of the transportation workflow.

Business Rule:

One order may generate one or multiple shipments depending on freight volume.

4. Dispatch Management Layer

DISPATCH

The **Dispatch** table manages transportation scheduling and execution.

Key attributes include:

- DispatchID (Primary Key)
- ShipmentID (Foreign Key)
- DispatchDate
- PlannedPickupDateTime
- PlannedDeliveryDateTime
- Status
- Notes
- VehicleID (Foreign Key)

Dispatch records coordinate operational scheduling.

Business Rule:

One shipment receives one dispatch assignment.

5. Driver Management Layer

DRIVER

The **Driver** table stores driver information.

Attributes include:

- DriverID (Primary Key)
- DriverName
- CDLNumber
- LicenseState
- HireDate

- Phone
- Email
- Status

This entity supports both employee and contractor drivers.

Future expansion opportunities:

- safety violations
- medical certifications
- hours-of-service compliance
- insurance tracking

6. Fleet Asset Management Layer

VEHICLE

The **Vehicle** Table stores trucks or tractors used for freight transportation.

Attributes include:

- VehicleID (Primary Key)
- UnitNumber
- VIN
- Year
- Make
- Model
- VehicleTypeID (Foreign Key)
- Status
- CurrentMileage

Business Rule:

Vehicles may participate in multiple dispatches over time.

VEHICLE TYPE

This lookup table standardizes vehicle classifications.

Examples:

- Day Cab
- Sleeper Cab
- Box Truck
- Refrigerated Truck

Attributes:

- VehicleTypeID
- TypeName
- Description
- DefaultCapacity

7. Trailer Management Layer

TRAILER

The **Trailer** table tracks trailer inventory.

Attributes include:

- TrailerID
- TrailerTypeID
- TrailerNumber
- VIN
- Capacity
- Status

TRAILER TYPE

This lookup table standardizes trailer classifications.

Examples:

- Dry Van

- Flatbed
- Refrigerated Trailer
- Tanker

Attributes include:

- TrailerTypeID
- TypeName
- Description
- DefaultCapacity

8. Route Planning Layer

ROUTE

The **Route** table stores transportation routes.

Attributes include:

- RouteID
- RouteName
- TotalDistance
- EstimatedDuration
- Description
- Status

Routes may be reused for recurring transportation lanes.

Examples:

- New York to Pennsylvania
- New Jersey to Ohio

9. Delivery Execution Layer

DELIVERY STOP

This table supports multi-stop routing.

Attributes include:

- StopID
- RouteID (Foreign Key)
- StopSequence
- StopType
- WarehouseID (Foreign Key)
- Address1
- Address2
- City
- State
- Zip
- ScheduledDateTime
- Notes

Business Rule:

One route may contain multiple delivery stops.

This allows:

- pickup stops
- drop-off stops
- transfer points
- cross-docking stops

10. Warehouse Operations Layer

WAREHOUSE

The **Warehouse** table tracks logistics facilities.

Attributes include:

- WarehouseID

- WarehouseName
- Type
- Address1
- Address2
- City
- State
- Zip
- Phone
- Status

Warehouses may serve as:

- origin locations
- destination locations
- temporary storage hubs
- distribution facilities

11. Fuel Cost Management Layer

FUEL TRANSACTION

Fuel costs represent one of the largest operational expenses in trucking.

This entity records:

- FuelTransactionID
- VehicleID (Foreign Key)
- DriverID (Foreign Key)
- TransactionDate
- OdometerReading
- Gallons
- PricePerGallon

- TotalCost
- Location

This table supports fuel efficiency reporting.

Example metrics:

- fuel cost per mile
 - cost per route
 - vehicle efficiency
-

12. Maintenance Management Layer

MAINTENANCE

This table tracks repairs and preventative maintenance.

Attributes include:

- MaintenanceID
- VehicleID (Foreign Key)
- MaintenanceDate
- MaintenanceTypeID (Foreign Key)
- Description
- Vendor
- Cost
- OdometerReading
- NextDueDate

MAINTENANCE TYPE

This lookup table standardizes maintenance categories.

Examples:

- Oil Change

- Tire Rotation
- Brake Repair
- Transmission Repair

Attributes:

- MaintenanceTypeID
- TypeName
- Description
- IntervalMiles
- IntervalDays

13. Billing Layer

INVOICE

The Invoice table manages customer billing.

Attributes include:

- InvoiceID
- CustomerID (Foreign Key)
- InvoiceDate
- DueDate
- TotalAmount
- Status

INVOICE LINE

This table supports detailed invoice breakdowns.

Attributes include:

- InvoiceLineID
- InvoiceID (Foreign Key)
- Description

- Quantity
- Rate
- LineAmount

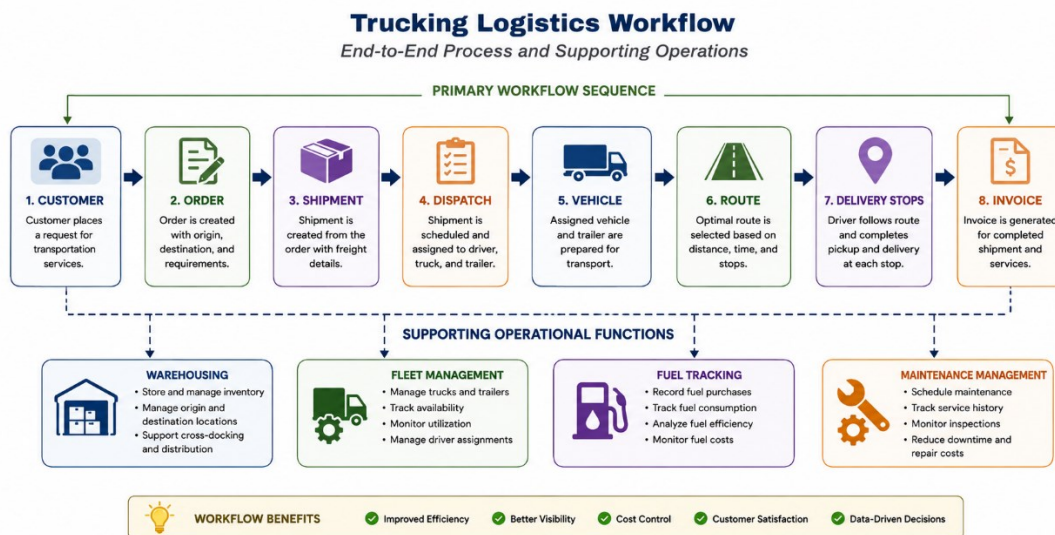
Examples:

- freight charge
- fuel surcharge
- detention fee
- special handling fee

End-to-End Business Workflow

The complete workflow supported by this logical model follows this sequence:

Customer → Order → Shipment → Dispatch → Vehicle → Route → Delivery Stops → Invoice



Supporting operational functions include:

- Warehousing
- Fleet Management
- Fuel Tracking
- Maintenance Management

Why This Logical Model Matters

This logical model provides a realistic framework for organizations that need to manage:

- freight transportation
- route planning
- fleet utilization
- customer billing
- operational cost analysis
- vehicle maintenance
- logistics reporting

It can serve as the foundation for:

- Transportation Management Systems (TMS)
- Fleet Management Platforms
- SQL Server training databases
- Data warehouse prototypes
- logistics analytics platforms
- educational demonstrations

The next phase of this downloadable package converts this logical model into fully functional **Microsoft SQL Server DDL scripts**, allowing users to physically implement the database.