**Overview of Payment Method Modules in Magento 2**

**Introduction:**

* Magento 2 is a robust e-commerce platform that offers various payment method modules to facilitate smooth transactions for merchants and customers.

**1. Magento\_Payment:**

* **Description:** This core payment module in Magento 2 handles fundamental payment functionalities such as credit card payments, checks, and money orders.

**2. Magento\_OfflinePayments:**

* **Description:** Provides support for offline payment methods including bank transfer, check/money order, and cash on delivery, allowing customers to complete transactions offline.

**3. Magento\_Braintree:**

* **Description:** Integrates the Braintree payment gateway into Magento 2, enabling merchants to accept payments via credit cards, PayPal, and other payment methods supported by Braintree.

**4. Magento\_Paypal:**

* **Description:** Integrates PayPal payment services into Magento 2, enabling merchants to accept payments via PayPal Express Checkout, PayPal Payments Standard, and PayPal Payflow Pro.

**5. Magento\_AdvancedPricingImportExport:**

* **Description:** Allows merchants to import and export advanced pricing settings, including payment-related configurations for products, aiding in managing pricing strategies efficiently.

**6. Magento\_Authorizenet:**

* **Description:** Integrates the Authorize.Net payment gateway into Magento 2, facilitating credit card payments and other payment methods supported by Authorize.Net.

**7. Magento\_Cybersource:**

* **Description:** Integrates the CyberSource payment gateway into Magento 2, enabling merchants to accept credit card payments and other payment methods supported by CyberSource.

**8. Magento\_InstantPurchase:**

* **Description:** Enables instant purchase functionality, allowing customers to skip the cart and proceed directly to checkout using stored payment and shipping information, enhancing the checkout experience.

**Creating payment method modules in Magento 2 core**

**Introduction:**

* Magento 2 provides a flexible architecture that allows developers to create custom payment method modules to integrate with various payment gateways or support specific payment workflows.

**1. Module Structure:**

* Create a new module directory under app/code following Magento's module structure conventions.
* Define module registration using registration.php.
* Create a module configuration file (module.xml) to declare module dependencies and version.

**2. Payment Method Implementation:**

* Create a payment method model class extending \Magento\Payment\Model\Method\AbstractMethod.
* Implement necessary methods such as authorize(), capture(), refund(), and void() according to the payment gateway's API.
* Define configuration options for the payment method in system.xml to allow configuration via admin panel.

**3. Payment Gateway Integration:**

* If integrating with a payment gateway, install the gateway's PHP SDK or API client library via Composer.
* Implement logic to communicate with the payment gateway's API for processing payments, refunds, and other operations.
* Handle response processing, error handling, and logging for transaction-related activities.

**4. Frontend Integration:**

* Implement frontend templates for checkout payment selection and additional form fields required by the payment method.
* Utilize JavaScript to handle client-side validation and dynamic behaviour of payment form fields.
* Ensure compatibility with Magento's Knockout.js-based checkout components for a seamless user experience.

**5. Backend Configuration:**

* Define backend configuration settings for the payment method in system.xml to specify credentials, API endpoints, and other configuration options.
* Implement backend logic to validate and save configuration settings entered by the admin user.

**6. Event Observers and Plugins:**

* Utilize event observers and plugins to hook into Magento's payment processing workflow and extend or modify behaviour as needed.
* Implement observers for events such as order placement, payment authorization, and order status updates.

**7. Testing:**

* Develop unit tests to ensure the correctness of payment method functionality and integration with payment gateways.
* Perform integration testing to verify end-to-end payment processing workflows in different scenarios.

**8. Packaging and Deployment**:

\* Package the module files.

\* Deploy the module to the Magento instance.

\*Verify proper installation and functionality.

**9. Maintenance and Updates**

\* Regularly update the module to ensure compatibility with new Magento versions and payment gateway updates.

\* Address any issues reported by users promptly.

**What are the tables used for saving the details into the database for example check and money order**

* After an order is placed using the Check/Money Order payment method in Magento 2, payment-related details are stored in the database.

**1. Order Table (sales\_order):**

* **Description:** The sales\_order table stores general information about each order in Magento 2.
* **Columns**

\*entity\_id: Primary key of the order record.

\*increment\_id: Unique increment ID assigned to the order.

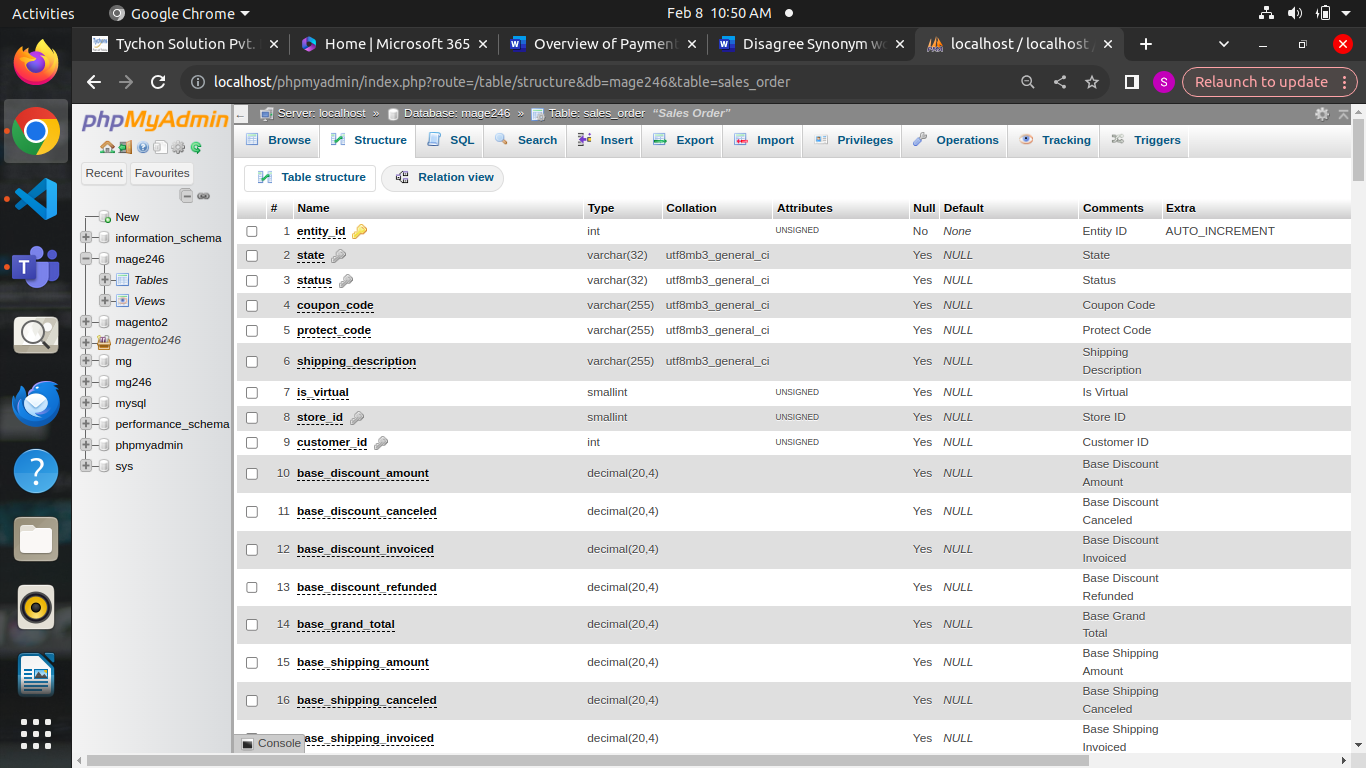
\*created\_at: Timestamp indicating the order creation date.

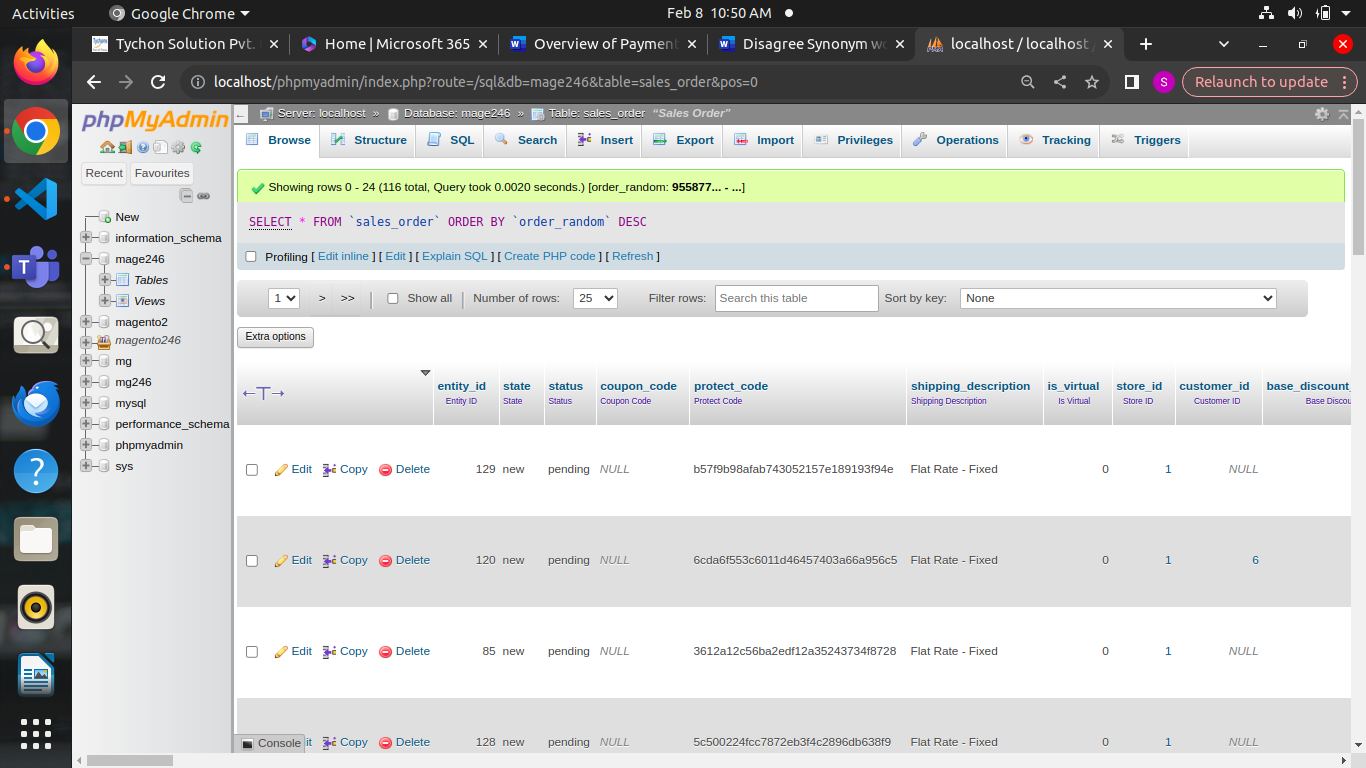
\*State: Current state of the order (e.g., new, processing, complete).

\*Status: status of the order. (status)

\*base\_grand\_total: Total amount of the order.

\*payment\_method: Payment method code used for the order (e.g., checkmo for Check/Money Order).





**2. Payment Information in Order Table:**

* When an order is placed using the Check/Money Order payment method, the payment details are saved directly in the sales\_order table.
* The relevant columns for payment information in the sales\_order table include:

\*base\_total\_due: Total amount due for the order. For Check/Money Order, this typically matches the order total.

\*total\_due: Total amount due in the order currency.

\*base\_total\_paid: Total amount paid for the order. Initially set to 0 for pending payments.

\*total\_paid: Total amount paid in the order currency.

\*base\_total\_paid\_real: Actual amount paid for the order. For Check/Money Order, this will be updated when the payment is received and processed.

\*total\_paid\_real: Actual amount paid in the order currency.

**3. Payment Processing Workflow:**

* After an order is placed using the Check/Money Order payment method, the order status typically remains as "pending" until payment is received and processed.
* When payment is received and processed, the payment details are updated in the sales\_order table, reflecting the actual amount paid (base\_total\_paid\_real and total\_paid\_real).
* The order status may be updated accordingly based on the payment status.

**After order placed with check/money order method.how payment related details saving on order tables**

**Introduction:**

* After an order is placed using the Check/Money Order payment method in Magento 2, payment-related details are stored in the database.

**1. Order Table (sales\_order):**

* **Description:** The sales\_order table stores general information about each order in Magento 2.
* **Columns of Interest:**

\*entity\_id: Primary key of the order record.

\*increment\_id: Unique increment ID assigned to the order.

\*created\_at: Timestamp indicating the order creation date.

\*state: Current state of the order (e.g., new, processing, complete).

\*Status: status of the order.

\*base\_grand\_total: Total amount of the order.

\*payment\_method: Payment method code used for the order (e.g., checkmo for Check/Money Order).

**2. Payment Information in Order Table:**

* When an order is placed using the Check/Money Order payment method, the payment details are saved directly in the sales\_order table.
* The relevant columns for payment information in the sales\_order table include:

\*base\_total\_due: Total amount due for the order. For Check/Money Order, this typically matches the order total.

\*total\_due: Total amount due in the order currency.

\*base\_total\_paid: Total amount paid for the order. Initially set to 0 for pending payments.

\*total\_paid: Total amount paid in the order currency.

\*base\_total\_paid\_real: Actual amount paid for the order. For Check/Money Order, this will be updated when the payment is received and processed.

\*total\_paid\_real: Actual amount paid in the order currency.

**3. Payment Processing Workflow:**

* After an order is placed using the Check/Money Order payment method, the order status typically remains as "pending" until payment is received and processed.
* When payment is received and processed, the payment details are updated in the sales\_order table, reflecting the actual amount paid (base\_total\_paid\_real and total\_paid\_real).
* The order status may be updated accordingly based on the payment status.

----------------------------------------------------------------------------------

**Tables related to payment**

**paypal\_payment\_transaction** :

* In Magento 2, the **paypal\_payment\_transaction** table is used to store information related to PayPal payment transactions. This table is part of the database schema used by Magento to manage payments made through PayPal.
* **transaction\_id**: This column stores a unique identifier for each PayPal transaction.
* **payment\_id**: This column typically stores the ID of the associated payment in Magento.
* **txn\_id**: This column stores the transaction ID provided by PayPal.
* **parent\_txn\_id**: If this transaction is a refund, this column stores the transaction ID of the original payment.
* **txn\_type**: This column stores the type of PayPal transaction, such as "payment", "authorization", "capture", "void", or "refund".
* **is\_closed**: This column indicates whether the transaction is closed or not.
* **additional\_information**: This column may store additional data related to the transaction in serialized or JSON format.
* **created\_at** and **updated\_at**: These columns store timestamps indicating when the transaction was created and last updated, respectively.

**Quote\_payment**

* In Magento 2, the **quote\_payment** table is used to store payment-related information during the checkout process. This table is part of the quote management system, which handles temporary data related to orders before they are completed.
* **entity\_id**: This column stores a unique identifier for each payment record.
* **quote\_id**: This column stores the ID of the associated quote, which represents the temporary order during the checkout process.
* **method**: This column stores the payment method code selected by the customer during checkout, such as "checkmo" for Check/Money Order, "paypal\_express" for PayPal Express Checkout, etc.
* **additional\_data**: This column may store additional data related to the payment method in serialized or JSON format, such as payment details, tokens, or extra information required by specific payment gateways.
* **cc\_type**: If the payment method involves credit card processing, this column stores the type of credit card used (e.g., Visa, MasterCard, etc.).
* **cc\_last4**: If the payment method involves credit card processing, this column typically stores the last four digits of the credit card number.
* **cc\_exp\_month** and **cc\_exp\_year**: If the payment method involves credit card processing, these columns store the expiration month and year of the credit card, respectively.
* **cc\_number\_enc**: If the payment method involves credit card processing and the platform supports it, this column may store an encrypted version of the credit card number for security purposes.
* **cc\_ss\_start\_month**, **cc\_ss\_start\_year**, **cc\_ss\_issue**: If the payment method involves credit card processing and the card is of the type that requires a start date or issue number (e.g., some Maestro cards), these columns may store that information.

**Sales\_order\_payment**

* In Magento 2, the **sales\_order\_payment** table is used to store payment-related information for completed orders. This table contains details about the payment method used, transaction details, and additional information related to the payment.
* **entity\_id:** This column stores a unique identifier for each payment record.
* **parent\_id:** This column stores the ID of the associated order in the sales\_order table.
* **method:** This column stores the payment method code used for the order, such as "checkmo" for Check/Money Order, "paypal\_express" for PayPal Express Checkout, etc.
* **additional\_information:** This column may store additional data related to the payment method in serialized or JSON format. It can include information such as payment details, transaction IDs, tokens, or any extra information required by specific payment gateways.
* **cc\_type:** If the payment method involves credit card processing, this column stores the type of credit card used (e.g., Visa, MasterCard, etc.).
* **cc\_last4:** If the payment method involves credit card processing, this column typically stores the last four digits of the credit card number.
* **cc\_exp\_month and cc\_exp\_year:** If the payment method involves credit card processing, these columns store the expiration month and year of the credit card, respectively.
* **cc\_number\_enc:** If the payment method involves credit card processing and the platform supports it, this column may store an encrypted version of the credit card number for security purposes.
* **cc\_ss\_start\_month, cc\_ss\_start\_year, cc\_ss\_issue:** If the payment method involves credit card processing and the card is of the type that requires a start date or issue number (e.g., some Maestro cards), these columns may store that information.
* **last\_trans\_id:** This column stores the transaction ID provided by the payment gateway for the order.
* **created\_at and updated\_at:** These columns store timestamps indicating when the payment record was created and last updated, respectively.

**Sales\_Payment\_transaction:**

* In Magento 2, the **sales\_payment\_transaction** table is used to store information about payment transactions associated with orders. This table plays a crucial role in tracking the payment history of orders and managing transactions.
* **transaction\_id**: This column stores a unique identifier for each payment transaction.
* **payment\_id**: This column stores the ID of the associated payment record in the **sales\_order\_payment** table.
* **order\_id**: This column stores the ID of the associated order in the **sales\_order** table.
* **txn\_id**: This column stores the transaction ID provided by the payment gateway for the transaction.
* **parent\_txn\_id**: If this transaction is related to another transaction (e.g., a refund or capture transaction related to an authorization), this column stores the ID of the parent transaction.
* **txn\_type**: This column stores the type of transaction, such as "payment", "authorization", "capture", "void", or "refund".
* **is\_closed**: This column indicates whether the transaction is closed or not.
* **additional\_information**: This column may store additional data related to the transaction in serialized or JSON format. It can include information such as payment details, response from the payment gateway, or any extra information specific to the transaction.
* **created\_at** and **updated\_at**: These columns store timestamps indicating when the transaction was created and last updated, respectively.

**Vault\_payment\_token**

In Magento 2, the **vault\_payment\_token** table is used to store information about payment tokens stored in the Magento Vault. The Magento Vault is a feature that securely stores sensitive payment information, such as credit card details, for future use, enabling customers to make purchases more conveniently without re-entering their payment information each time.

* **entity\_id**: This column stores a unique identifier for each payment token record.
* **customer\_id**: This column stores the ID of the associated customer in the **customer\_entity** table.
* **public\_hash**: This column stores a hash value that uniquely identifies the payment token publicly.
* **payment\_method\_code**: This column stores the payment method code associated with the payment token.
* **created\_at** and **updated\_at**: These columns store timestamps indicating when the payment token was created and last updated, respectively.
* **expires\_at**: This column stores the expiration date of the payment token, if applicable.
* **gateway\_token**: This column stores the token provided by the payment gateway, which is used to reference the payment method and initiate transactions securely.

**Vault\_payment\_token\_order\_payment\_link**

In Magento 2, the **vault\_payment\_token\_order\_payment\_link** table serves as a link between payment tokens stored in the Magento Vault and orders. This table is used to associate payment tokens with specific orders, enabling Magento to process payments using the stored tokens.

* **link\_id**: This column stores a unique identifier for each link record.
* **token\_id**: This column stores the ID of the associated payment token in the **vault\_payment\_token** table.
* **order\_id**: This column stores the ID of the associated order in the **sales\_order** table.
* **payment\_id**: This column stores the ID of the associated payment record in the **sales\_order\_payment** table.
* **created\_at** and **updated\_at**: These columns store timestamps indicating when the link record was created and last updated, respectively.

**Payment modules file structure in Magento 2**

* Magento 2 provides a flexible and modular architecture for implementing payment gateways.
* Payment modules in Magento 2 follow a specific file structure to ensure consistency and ease of development.

## **File Structure Details**

### **1. Block/**

Contains PHP classes responsible for presenting data in the frontend.

### **2. Controller/**

Contains PHP classes handling payment-related actions, such as processing callbacks.

### **3. etc/**

Contains configuration files for the module.

* adminhtml/system.xml: Configuration for backend settings.
* frontend/di.xml: Dependency injection configuration for frontend.
* module.xml: Module configuration file.

### **4. Model/**

Contains PHP classes responsible for business logic related to payments.

### **5. Observer/**

Contains PHP classes observing payment-related events

### **6. view/**

Contains layout and template files for the frontend and backend.

* adminhtml/templates/payment/form.phtml: Backend template for payment form.
* frontend/templates/payment/form.phtml: Frontend template for payment form.

### **7. registration.php**

Registers the module with Magento.

### **8. design/frontend/VendorName/ThemeName/layout/checkout\_index\_index.xml**

Layout XML file to customize the checkout page for the payment method.

**Payments related events:**

* **payment\_method\_is\_active**: This event is dispatched to determine whether a payment method is active and available for use during the checkout process.
* **payment\_method\_assign\_data**: This event is dispatched when data is assigned to a payment method during the checkout process. Developers can use this event to manipulate or validate payment method data before it is saved.
* **payment\_method\_import\_data**: This event is dispatched when payment method data is imported during the checkout process.
* **payment\_method\_prepare**: This event is dispatched before the payment method is prepared for use during the checkout process.
* **payment\_method\_list**: This event is dispatched to modify the list of available payment methods displayed during the checkout process.
* **payment\_method\_render**: This event is dispatched to render the payment method form displayed during the checkout process.
* **payment\_method\_save**: This event is dispatched after the payment method is saved during the checkout process.
* **sales\_order\_payment\_place\_start**: This event is dispatched when the payment process for an order starts.
* **sales\_order\_payment\_place\_end**: This event is dispatched when the payment process for an order end.
* **sales\_order\_payment\_pay**: This event is dispatched when a payment is made for an order.
* **sales\_order\_payment\_refund**: This event is dispatched when a refund is processed for an order.
* **sales\_order\_payment\_void**: This event is dispatched when a payment authorization is voided for an order.
* **sales\_order\_payment\_capture**: This event is dispatched when a payment authorization is captured for an order.
* **sales\_order\_payment\_cancel**: This event is dispatched when a payment is canceled for an order.