Product-Customer Matrix Prices

PDF Manual

MODULE

productcustomermatrix

LAST UPDATED

2026-03-29

Matrix Pricing System

Product-Customer Matrix Prices · /productcustomermatrix/features/matrix-pricing

Deep dive into the core matrix pricing architecture, workflow, and integration with Magento 2 Pricesystem.

Overview

Matrix Pricing enables sophisticated product-customer price matching through configurable matrices that act as logical pricing containers. Unlike traditional pricelists where you assign customers to lists, matrices use **bidirectional matching**: products can be matched to customer attributes, or customer attributes can select products.

Key Concepts

Matrix Container: Logical unit that groups:

- Product selection rules
- Customer matching rules
- Priority for conflict resolution
- Date range validity
- Quantity-based pricing tiers

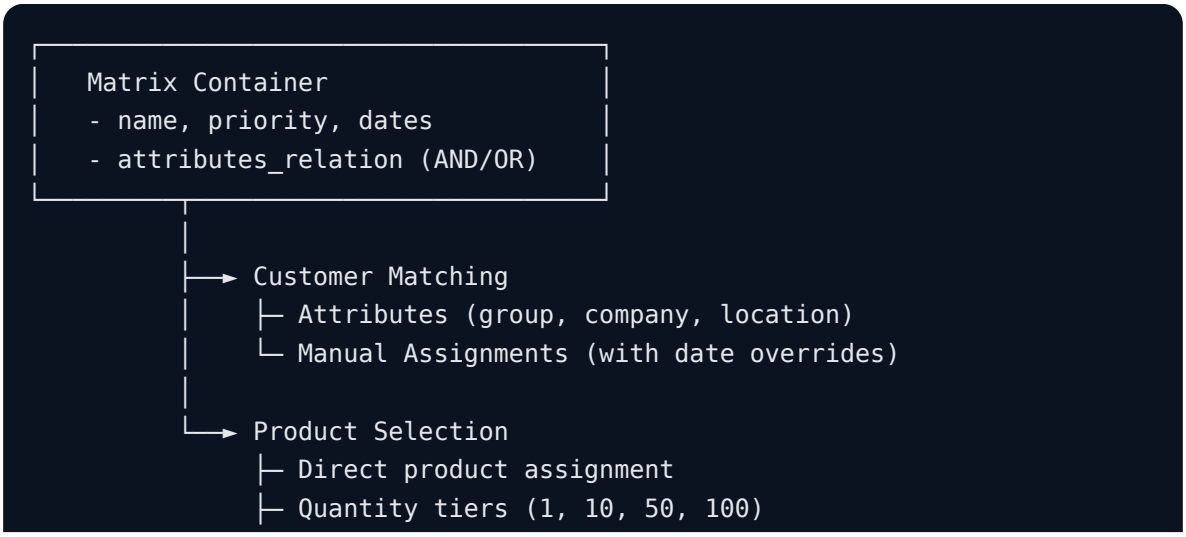
Bidirectional Matching:

- **Traditional:** Customer → Pricelist → Products
- **Matrix:** Customer ←→ Attributes ←→ Products

Attribute-Based Logic: Customers matched via attributes (group, company, location) rather than manual assignment.

Matrix Architecture

Component Diagram



```
                └─ Product-level date ranges
```

Data Flow

```
Customer Login
    ↓
Evaluate Active Matrices
    ↓
Check Attributes Match (AND/OR logic)
    ↓
Filter by Date Validity (matrix + customer dates)
    ↓
Assign Matching Matrices
    ↓
Cache Assignment (session)
    ↓
Product View/Cart
    ↓
Resolve Price (priority + merge logic)
    ↓
Display Final Price
```

Matrix Lifecycle

1. Creation Phase

Admin Creates Matrix:

- Set basic info (name, priority, dates)
- Configure attributes (group, company, location)
- Choose attributes relation (AND/OR)
- Save matrix container

Status: Inactive (default) or Active

2. Product Assignment

Add Products:

- Select products from catalog
- Set prices for each product
- Add quantity tiers (optional)
- Set product-level date ranges (optional)

Example:

```
Product: Widget Pro (SKU-123)
Tiers:
  - qty=1, price=$100, dates=NULL
```

```
- qty=10, price=$95, dates=NULL
- qty=50, price=$90, dates=2025-06-01 to 2025-08-31 (summer special)
```

3. Customer Assignment

Automatic (Attribute-Based):

- System evaluates customer attributes on login
- Matches against matrix attribute rules
- Auto-assigns if all conditions met

Manual:

- Admin explicitly assigns customer to matrix
- Optional: Set customer-specific date overrides
- Bypasses attribute matching

4. Activation

Set Matrix Active:

- `is_active = Yes`
- Matrix immediately available for matching

Date-Based Activation:

- `from_date = 2025-06-01`
- Automatically activates June 1st
- No manual intervention needed

5. Price Resolution

Customer Views Product:

1. System finds all assigned matrices
2. Filters by date validity
3. Checks product exists in matrix
4. Sorts by priority (highest first)
5. Applies merge logic (if enabled)
6. Returns final price for current quantity

6. Deactivation/Expiration

Manual Deactivation:

- Set `is_active = No`
- Immediately stops price evaluation

Automatic Expiration:

- `to_date = 2025-12-31`
- Expires January 1, 2026
- No cleanup needed

Price Resolution Algorithm

Step-by-Step Resolution

Input:

- Customer ID: 123
- Product ID: 456
- Quantity: 25

Process:

Step 1: Find Matching Matrices

```
SELECT m.*
FROM pricesystem_product_customer_matrix m
WHERE m.is_active = 1
  AND m.website_id = 1
  AND (m.from_date IS NULL OR m.from_date <= CURDATE())
  AND (m.to_date IS NULL OR m.to_date >= CURDATE())
  AND m.id IN (
    -- Matrices customer is assigned to
    SELECT matrix_id FROM pricesystem_product_customer_matrix_customer
    WHERE customer_id = 123
  )
ORDER BY m.priority DESC;
```

Result: 3 matrices (A=Priority 15, B=Priority 20, C=Priority 30)

Step 2: Check Product Availability

```
SELECT p.*
FROM pricesystem_pricelist_product p
WHERE p.pricelist_id IN (matrixA, matrixB, matrixC)
  AND p.product_id = 456
  AND (p.from_date IS NULL OR p.from_date <= CURDATE())
  AND (p.to_date IS NULL OR p.to_date >= CURDATE());
```

Result: Product exists in all 3 matrices

Step 3: Find Quantity Tiers

```
SELECT p.qty, p.price, p.pricelist_id
FROM pricesystem_pricelist_product p
WHERE p.pricelist_id IN (matrixA, matrixB, matrixC)
  AND p.product_id = 456
  AND p.qty <= 25  -- Customer quantity
ORDER BY p.pricelist_id, p.qty DESC;
```

Result:

```
Matrix A (Priority 15):
  - qty=1: $100
  - qty=10: $95
  - qty=25: $92  ← Best tier for qty=25

Matrix B (Priority 20):
  - qty=1: $98
  - qty=10: $93
  (no qty=25 tier, use qty=10)

Matrix C (Priority 30):
  - qty=1: $96
  - qty=50: $88
  (no qty=25 tier, use qty=1)
```

Step 4: Apply Merge Logic

If merge_matrix_qtys = No (Highest Priority Only):

```
Use Matrix C only (Priority 30)
Product qty=25 → Use Matrix C qty=1 tier: $96
Result: $96/unit
```

If merge_matrix_qtys = Yes (Best Price Wins):

```
Compare best price at qty=25 across all matrices:
- Matrix A qty=25: $92  ← WINNER
- Matrix B qty=10: $93
- Matrix C qty=1: $96

Result: $92/unit (from Matrix A)
```

Step 5: Return Final Price

Output:

- merge=No: $96/unit × 25 = $2,400 total
- merge=Yes: $92/unit × 25 = $2,300 total (customer saves $100!)

Integration with Pricesystem Core

Price Resolution Order (Full System)

Product-Customer Matrix integrates into Magento 2 Pricesystem price resolution:

```
1. Shopping Cart Price Rules (if enabled)
   ↓
2. Customer-Specific Prices (direct assignments)
   ↓
3. Product-Customer Matrices ← THIS EXTENSION
   ↓
4. Named Pricelists (pricesystem-pricelist)
   ↓
5. Category Prices (pricesystem-categoryprice)
   ↓
6. Catalog Price (fallback)
```

Example Scenario:

```
Product: Widget Pro
Catalog Price: $150

Customer has:
- Matrix pricing: $100 (Priority 20)
- Pricelist pricing: $110 (Priority 15)
- Category pricing: $120

Resolution:
Step 1: No cart rules
Step 2: No customer-specific price
Step 3: Matrix price found: $100 ← WINNER
(Pricelists and category prices never evaluated)

Final Price: $100
```

Cross-Module Conflicts

Matrix vs Pricelist (both active):

- Matrices evaluated BEFORE pricelists
- Matrix price found → Pricelist ignored
- No matrix price → Fallback to pricelist

Matrix vs Category Price:

- Matrices evaluated BEFORE category prices
- Same logic as pricelists

Priority Within Matrices:

- Multiple matrices resolved via priority field
- NOT via module loading order

Advanced Scenarios

Scenario 1: Overlapping Date Ranges

Setup:

```
Matrix A: "Standard Wholesale" (Priority 15)
- Dates: 2025-01-01 to 2025-12-31 (permanent)
- Product X: $100

Matrix B: "Black Friday Special" (Priority 25)
- Dates: 2025-11-29 to 2025-12-02 (4 days)
- Product X: $75
```

Timeline:

- **Nov 28:** Only Matrix A active → $100
- **Nov 29-Dec 2:** Both active, Matrix B wins (Priority 25) → $75
- **Dec 3+:** Only Matrix A active → $100

Key Point: Temporary high-priority matrices override standard pricing.

Scenario 2: Customer-Specific Date Override

Setup:

```
Matrix: "Annual Contract - ACME Corp" (Priority 35)
- Dates: 2025-01-01 to 2025-12-31
- Company: "ACME"
- Product X: $90

Customer #123 (ACME employee):
- Manual assignment to matrix
- Override dates: 2025-01-01 to 2025-06-30 (6-month trial)

Customer #456 (ACME employee):
- Automatic assignment via company match
- No override dates (uses matrix dates)
```

Result:

- Customer #123: Sees $90 from Jan 1 to Jun 30, then loses access
- Customer #456: Sees $90 from Jan 1 to Dec 31 (full year)

Use Case: Trial periods, temporary access, phased rollouts.

Scenario 3: Product-Level Date Override

Setup:

```
Matrix: "Seasonal Pricing 2025" (Priority 20)
- Dates: 2025-01-01 to 2025-12-31 (permanent)
- Customer Group: Wholesale

Product X:
- qty=1: $100, dates=NULL (year-round)
- qty=10: $95, dates=NULL (year-round)
- qty=50: $85, dates=2025-06-01 to 2025-08-31 (summer special)
```

Result:

- Jan-May: qty tiers: 1=$100, 10=$95 (qty=50 tier not available)
- Jun-Aug: qty tiers: 1=$100, 10=$95, 50=$85 (summer tier active)
- Sep-Dec: qty tiers: 1=$100, 10=$95 (qty=50 tier expired)

Use Case: Seasonal quantity discounts, limited-time bulk pricing.

Scenario 4: Multi-Product Matrix with Partial Overlap

Setup:

```
Matrix A (Priority 15): Products X, Y, Z
Matrix B (Priority 20): Products X, Y (Z missing)

Customer matches both matrices.
merge_matrix_qtys = No (highest priority only)
```

Result:

- Product X: Uses Matrix B (Priority 20)
- Product Y: Uses Matrix B (Priority 20)
- Product Z: **NO MATRIX PRICE** (Matrix B doesn't have Z, Matrix A ignored)
    - Fallback to next pricesystem module (pricelist, category, catalog)

Lesson: High-priority matrices with incomplete product sets can cause fallback.

Solution: Use `merge_matrix_qtys = Yes` to include Matrix A prices for Product Z.

Scenario 5: Three-Way Priority Resolution

Setup:

```
Customer: John Doe
```

```
Matches three matrices:

Matrix A: "Wholesale 2025" (Priority 15)
- Product X: qty=1 ($100), qty=10 ($95), qty=50 ($90)

Matrix B: "California Regional" (Priority 20)
- Product X: qty=1 ($98), qty=25 ($92)

Matrix C: "ACME Contract" (Priority 30)
- Product X: qty=1 ($96), qty=100 ($88)
```

Customer orders 30 units of Product X.

merge_matrix_qtys = No:

```
Use Matrix C only (Priority 30)
Best tier for qty=30: qty=1 ($96)
Result: $96/unit × 30 = $2,880
```

merge_matrix_qtys = Yes:

```
Merge all tiers:
- qty=1: Best of ($100, $98, $96) = $96 (Matrix C)
- qty=10: Best of ($95, $98, $96) = $95 (Matrix A)
- qty=25: Best of ($90, $92, $96) = $90 (Matrix A)
- qty=50: Best of ($90, $92, $96) = $90 (Matrix A)
- qty=100: $88 (Matrix C)

Customer orders 30 units → Uses qty=25 tier: $90
Result: $90/unit × 30 = $2,700

Savings: $180 by using merge!
```

Database Operations

Creating a Matrix

```
INSERT INTO pricesystem_product_customer_matrix
(name, is_active, priority, from_date, to_date, website_id,
attributes_relation, created_at, updated_at)
VALUES
('Wholesale US 2025', 1, 15, '2025-01-01', '2025-12-31', 1, 'AND', NOW(),
NOW());

SET @matrix_id = LAST_INSERT_ID();
```

Adding Attributes

```
INSERT INTO pricesystem_product_customer_matrix_attribute
(matrix_id, attribute_code, attribute_value)
VALUES
(@matrix_id, 'group', '2'),  -- Wholesale group
(@matrix_id, 'country', 'US');  -- United States
```

Adding Products with Qty Tiers

```
INSERT INTO pricesystem_pricelist_product
(pricelist_id, product_id, qty, price, from_date, to_date)
VALUES
(@matrix_id, 123, 1, 100.00, NULL, NULL),
(@matrix_id, 123, 10, 95.00, NULL, NULL),
(@matrix_id, 123, 50, 90.00, NULL, NULL),
(@matrix_id, 123, 100, 85.00, NULL, NULL);
```

Manual Customer Assignment

```
INSERT INTO pricesystem_product_customer_matrix_customer
(matrix_id, customer_id, from_date, to_date)
VALUES
(@matrix_id, 456, '2025-01-01', '2025-06-30');  -- 6-month trial
```

Finding Customer's Matrices

```
SELECT
  m.id,
  m.name,
  m.priority,
  m.from_date,
  m.to_date,
  CASE
    WHEN mc.from_date IS NOT NULL THEN mc.from_date
    ELSE m.from_date
  END AS effective_from_date,
  CASE
    WHEN mc.to_date IS NOT NULL THEN mc.to_date
    ELSE m.to_date
  END AS effective_to_date
FROM pricesystem_product_customer_matrix m
JOIN pricesystem_product_customer_matrix_customer mc ON mc.matrix_id = m.id
WHERE mc.customer_id = 123
  AND m.is_active = 1
ORDER BY m.priority DESC;
```

Performance Optimization

Caching Strategy

Matrix Assignment Cache:

- Cached per customer session
- Cache key: `matrix_assignment_customer_{id}`
- TTL: Session lifetime
- Invalidated on: Profile update, group change, logout

Price Cache:

- Cached per customer + product + qty
- Cache key: `matrix_price_{customer_id}_{product_id}_{qty}`
- TTL: Configurable (default: 1 hour)
- Invalidated on: Matrix update, product update, price change

Indexing

Critical Indexes:

```
-- Matrix priority + active status
CREATE INDEX idx_matrix_active_priority ON
pricesystem_product_customer_matrix(is_active, priority);

-- Matrix dates
CREATE INDEX idx_matrix_dates ON
pricesystem_product_customer_matrix(from_date, to_date);

-- Customer assignments
CREATE INDEX idx_matrix_customer ON
pricesystem_product_customer_matrix_customer(customer_id);

-- Product prices
CREATE INDEX idx_product_matrix_qty ON
pricesystem_pricelist_product(pricelist_id, product_id, qty);
```

Query Optimization

Avoid:

```
-- BAD: Scans all matrices for each customer
SELECT * FROM pricesystem_product_customer_matrix WHERE is_active = 1;
```

Prefer:

```
-- GOOD: Uses customer assignment index
```

```
SELECT m.*
FROM pricesystem_product_customer_matrix m
JOIN pricesystem_product_customer_matrix_customer mc ON mc.matrix_id = m.id
WHERE mc.customer_id = 123
  AND m.is_active = 1;
```

Best Practices

Matrix Design

1. **Use specific attributes:** Group + Country faster than Company matching
2. **Leave gaps in priority:** 10, 20, 30 (not 10, 11, 12) allows insertions
3. **Document hierarchy:** Team reference for priority tiers
4. **Test edge cases:** Overlapping dates, missing products, same priority

Quantity Tiers

1. **Consistent tiers:** Use same breakpoints across products (1, 10, 50, 100)
2. **Logical progression:** Each tier should offer meaningful discount
3. **Avoid too many tiers:** 4-6 tiers maximum (performance + UX)
4. **Document strategy:** Why these breakpoints?

Date Management

1. **Include grace periods:** Contract renewals need overlap
2. **Test transitions:** Verify date changes in staging
3. **Use future dates:** Schedule campaigns in advance
4. **Monitor expirations:** Alert before critical matrices expire

Assignment Strategy

1. **Prefer automatic:** Attribute-based assignment scales better
2. **Manual for exceptions:** Override automatic rules sparingly
3. **Document overrides:** Why this customer is special?
4. **Regular audits:** Review manual assignments quarterly

Troubleshooting

Debug Checklist

Matrix not applying:

1. ✓ Matrix active? (`is_active = 1`)
2. ✓ Dates valid? (matrix + customer override)
3. ✓ Customer assigned? (check both automatic + manual)
4. ✓ Product exists in matrix?
5. ✓ Quantity tier exists for cart quantity?

6. ✓ Priority conflicts? (other matrix winning)
7. ✓ Cache cleared?

Wrong price displaying:

1. ✓ Check merge config (`merge_matrix_qtys`)
2. ✓ Verify priority values (highest should win)
3. ✓ Check quantity tier (correct tier for cart quantity?)
4. ✓ Review other pricesystem modules (higher precedence?)
5. ✓ Inspect cache (stale price cached?)

Related Documentation

- Creating Matrices - Step-by-step creation guide
- Multi-Attribute Matching - Attribute logic with AND/OR
- Priority Resolution - Conflict resolution strategies