Relaciones entre Entidades - SIESA ERP ↔ Kong WMS/RFID

Diagrama de Relaciones Generales

mermaid

```
erDiagram
    SIESA_ITEMS ||--o{ KONG_SKUS : sincroniza
    SIESA_TERCEROS ||--o{ KONG_CUSTOMERS : sincroniza
    SIESA_BODEGAS ||--o{ KONG_LOCATIONS : sincroniza

    SIESA_ORDENES_COMPRA ||--o{ KONG_PURCHASE_ORDERS : sincroniza
    SIESA_ORDENES_VENTA ||--o{ KONG_STORE_ORDERS : sincroniza

    KONG_MOVES ||--|| SIESA_DOCUMENTO_INV : "genera entrada/salida"
    KONG_AUDITS ||--o{ SIESA_DOCUMENTO_INV : "genera ajustes"
    KONG_PACKINGS ||--|| SIESA_DOCUMENTO_INV : "genera salida"

    KONG_SKUS ||--o{ KONG_ITEMS : "instancia fisica"
    KONG_LOCATIONS ||--o{ KONG_ITEMS : "contiene"
    KONG_MOVES ||--o{ KONG_MOVE_LINES : "detalle"
    KONG_STORE_ORDERS ||--o{ KONG_STORE_ORDER_LINES : "detalle"
```

Relación 1: Productos (SIESA → Kong)

Tipo de Relación

- **Cardinalidad**: 1:1 (Uno a Uno)
- **Dirección**: SIESA → Kong
- **Obligatoriedad**: Obligatoria

- **Integridad Referencial**: Cascade (actualización)

Entidades Involucradas

- **SIESA**: Items/Productos (f120_*)
- **Kong**: SKUs (inventory/skus/)

Campos de Unión

Entidad SIESA	Campo	Entidad Kong	Campo
Items	f120_referencia	SKUs	external_id

Representación

```javascript
// Un producto SIESA = Un SKU Kong
SIESA_Item {
  f120_referencia: "PROD-001"
} → Kong_SKU {
  external_id: "PROD-001"
}
```

Reglas de Sincronización

1. **Creación**:
   - Al crear producto en SIESA → Crear SKU en Kong
   - Validar que external_id no exista antes de crear
2. **Actualización**:
   - Cambios en SIESA → Actualizar SKU en Kong
   - Usar external_id para localizar el SKU

3. **Eliminación/Inactivación**:
   - Producto inactivo en SIESA → is_active = false en Kong
   - NO eliminar físicamente, solo marcar inactivo

Ejemplo de Sincronización

```javascript
// Sincronizar producto SIESA → Kong
async function sincronizarProducto(productoSIESA) {
  // 1. Buscar si existe en Kong
  const skuExistente = await buscarSKUKong(productoSIESA.f120_referencia);

  const skuData = transformarProductoSIESAaKong(productoSIESA);

  if (skuExistente) {
    // 2. Actualizar
    await actualizarSKUKong(skuExistente.id, skuData);
  } else {
    // 3. Crear nuevo
    await crearSKUKong(skuData);
  }
}
```

Relación 2: SKU → Items (Instancias Físicas en Kong)

Tipo de Relación

- **Cardinalidad**: 1:N (Uno a Muchos)
- **Dirección**: Kong interno

- **Obligatoriedad**: Opcional
- **Integridad Referencial**: Cascade

Entidades Involucradas

- **Kong Master**: SKU (plantilla)
- **Kong Detalle**: Item (instancia física con EPC)

Campos de Unión

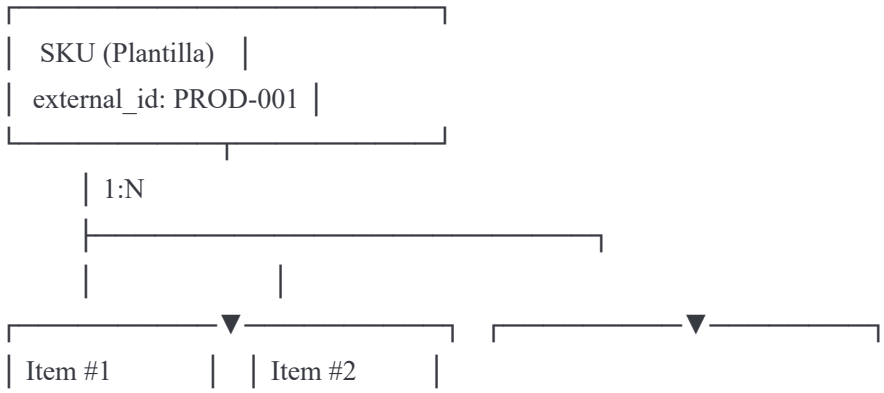
Entidad Padre	Campo	Entidad Hija	Campo
SKU	id	Item	sku_id

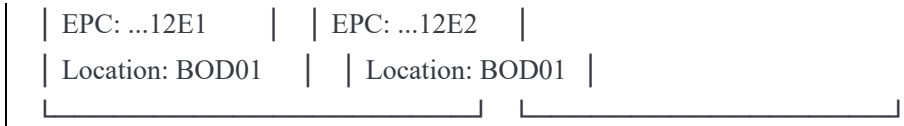
Representación

javascript

```
// Un SKU puede tener múltiples Items (instancias físicas)
Kong_SKU {
  id: 456,
  external_id: "PROD-001"
}
  ↓ tiene múltiples
Kong_Items [
  {
    id: 1001,
    sku_id: 456,
    epc: "E280116060000020942A12E1",
    location_id: 10
  },
  {
    id: 1002,
    sku_id: 456,
    epc: "E280116060000020942A12E2",
    location_id: 10
  }
]
```

Diagrama Detallado





Reglas de Negocio

1. **Creación de Items**:
   - Ocurre durante recepción con RFID
   - Cada EPC único genera un Item
   - Item debe vincularse a un SKU existente
2. **Movimiento de Items**:
   - Items cambian de location al moverse físicamente
   - RFID rastrea cada Item individual
3. **Consulta de Inventario**:
   - Inventario = COUNT(Items) GROUP BY SKU
   - No se reportan Items individuales a SIESA, solo totales por SKU

Relación 3: Órdenes de Compra (SIESA → Kong)

Tipo de Relación

- **Cardinalidad Cabecera**: 1:1
- **Cardinalidad Detalle**: 1:N
- **Dirección**: SIESA → Kong
- **Obligatoriedad**: Opcional (solo si se usa recepción con OC)

Entidades Involucradas

- **SIESA Cabecera**: Orden de Compra (f430_*)
- **SIESA Detalle**: Líneas OC (f431_*)
- **Kong Cabecera**: Purchase Order
- **Kong Detalle**: Purchase Order Lines

Campos de Unión

Cabecera

SIESA	Campo	Kong	Campo
Orden Compra	f430_consec_docto	Purchase Order	external_id

Detalle

SIESA	Campo	Kong	Campo
Línea OC	f431_id_item	PO Line	sku_external_id
Línea OC	f431_nro_registro	PO Line	line_number

Representación

javascript

```
SIESA_OrdenCompra {
  f430_consec_docto: "OC-12345",
  f430_id_tercero: "PROV001",
  lineas: [
    { f431_id_item: "PROD-001", f431_cantidad: 50 },
    { f431_id_item: "PROD-002", f431_cantidad: 100 }
  ]
}
  ↓ sincroniza
Kong_PurchaseOrder {
  external_id: "OC-12345",
  requester_external_id: "PROV001",
  lines: [
    { sku_external_id: "PROD-001", quantity: 50 },
    { sku_external_id: "PROD-002", quantity: 100 }
  ]
}
```

Flujo de Datos

```
1. SIESA crea OC
   ↓
2. Sincronizar a Kong (Purchase Order)
   ↓
3. Operador recibe mercancía (Kong Move)
   ↓
4. Al cerrar Move → Generar Entrada en SIESA
   ↓
5. SIESA actualiza estado OC (parcial/completa)
```

Relación 4: Movimientos de Inventario (Kong → SIESA)

Tipo de Relación

- **Cardinalidad Cabecera**: N:1 (Muchos Kong Moves → Muchos Doc SIESA)
- **Cardinalidad Detalle**: 1:N (Un Move → N líneas)
- **Dirección**: Kong → SIESA
- **Obligatoriedad**: Obligatoria

Entidades Involucradas

- **Kong Cabecera**: Move
- **Kong Detalle**: Move Lines
- **SIESA Cabecera**: Documento Inventario (f350_*)
- **SIESA Detalle**: Movimientos (f470_*)

Campos de Unión

Referencia Cruzada

Kong	Campo	SIESA	Campo
Move	id	Documento	f450_docto_alterno

Representación

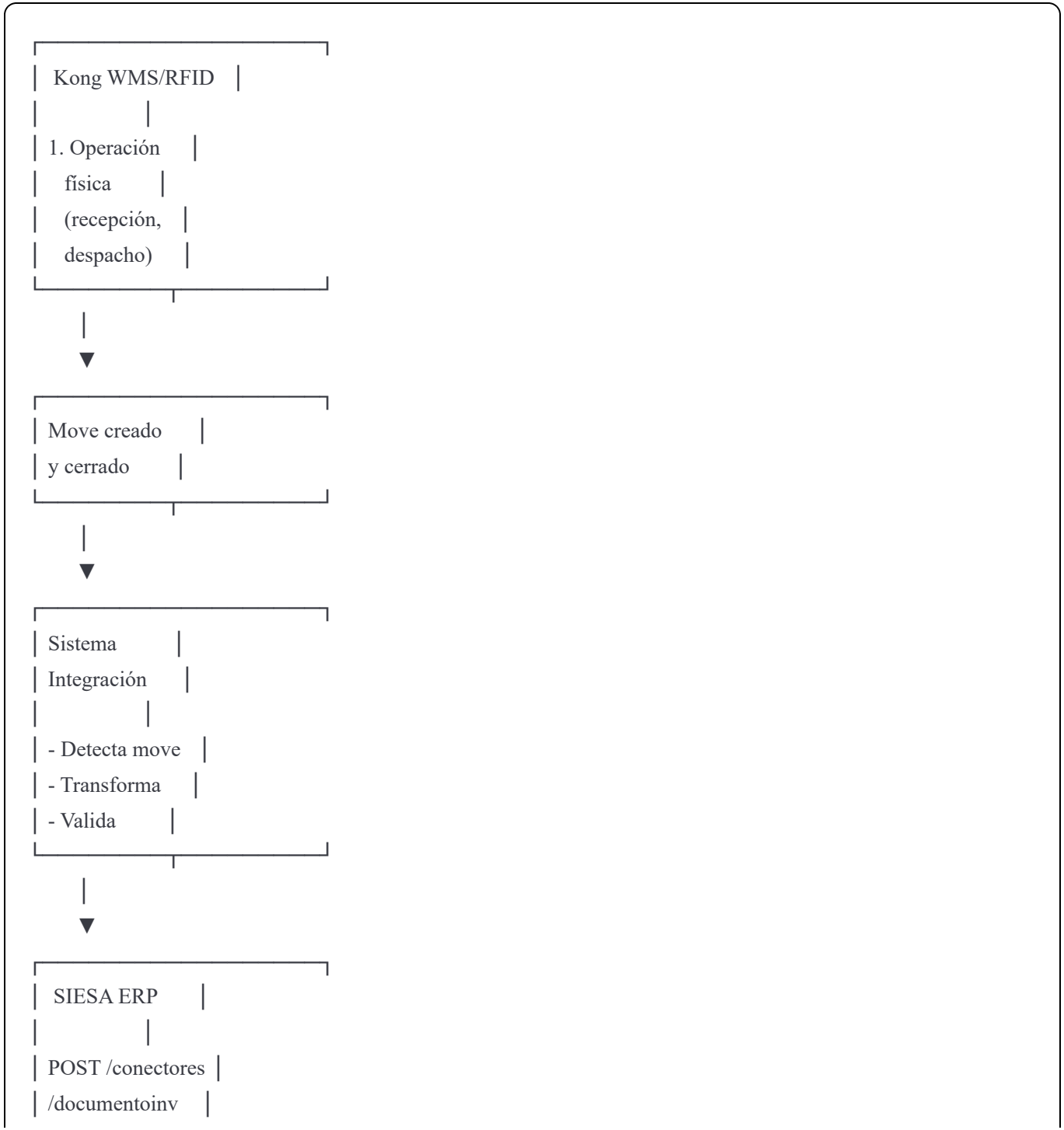
javascript

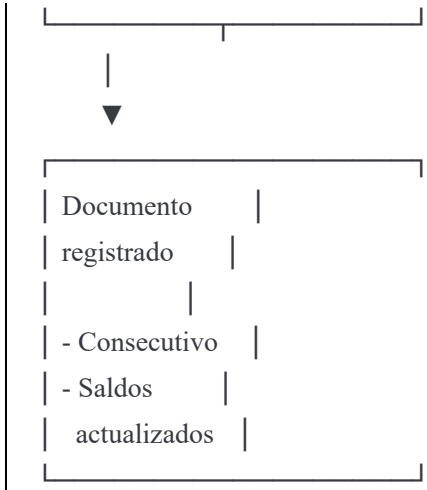
```
Kong_Move {
  id: 789,
  move_type: "RECEIVING",
  destination_location: { external_id: "BOD01" },
  lines: [
    { sku: { external_id: "PROD-001" }, quantity_received: 50 },
    { sku: { external_id: "PROD-002" }, quantity_received: 25 }
  ]
}
  ↓ genera
SIESA_Documento {
  f350_id_tipo_docto: "ENT",
  f450_id_bodega_entrada: "BOD01",
  f450_docto_alterno: "KONG-MOVE-789",
  Movimientos: [
    { f470_id_item: "PROD-001", f470_cant_base: "50" },
    { f470_id_item: "PROD-002", f470_cant_base: "25" }
  ]
}
```

Mapeo de Tipos

Kong Move Type	SIESA Tipo Docto	SIESA Concepto
RECEIVING	ENT	1
SHIPPING	SAL	2
TRANSFER	TRA	5
ADJUSTMENT (positivo)	AJU	3
ADJUSTMENT (negativo)	AJU	4

Diagrama de Flujo Completo





Relación 5: Auditorías y Ajustes

Tipo de Relación

- **Cardinalidad**: N:M (Muchas líneas de auditoría → Múltiples ajustes)
- **Dirección**: Kong → SIESA
- **Obligatoriedad**: Condicional (solo si hay diferencias)

Entidades Involucradas

- **Kong**: Audit, Audit Lines
- **SIESA**: Documentos de Ajuste (AJU)

Flujo de Ajuste

1. Kong Audit creada
   ↓
2. Conteo físico con RFID
   ↓

3. Calcular diferencias (físico vs sistema Kong)
   ↓
4. Para cada SKU con diferencia:
   ├─ Consultar saldo SIESA
   ├─ Calcular diferencia (físico vs SIESA)
   └─ Si diferencia ≠ 0:
      └─ Generar ajuste SIESA

Ejemplo de Transformación

javascript

```
// Línea de auditoría con diferencia
Kong_AuditLine {
  audit: 101,
  sku: { external_id: "PROD-001" },
  location: { external_id: "BOD01" },
  physical_quantity: 93,  // Conteo RFID
  system_quantity: 95     // Sistema Kong
}

// Consultar SIESA
Saldo_SIESA {
  item: "PROD-001",
  bodega: "BOD01",
  saldo_cantidad: 95      // Saldo contable
}

// Generar ajuste
Diferencia = 93 - 95 = -2 (Faltante)

SIESA_Ajuste {
  f350_id_tipo_docto: "AJU",
  f450_id_concepto: "4",  // Ajuste negativo
  f450_id_bodega_salida: "BOD01",
  Movimientos: [{
    f470_id_item: "PROD-001",
    f470_cant_base: "2"
  }]
}
```

Relación 6: Ubicaciones y Saldos

Estructura Jerárquica en Kong

```
Location (Bodega)
  ├─ Sub-Location (Zona)
  │   ├─ Sub-Location (Rack)
  │   │   └─ Items
  │   └─ Sub-Location (Rack)
  │       └─ Items
  └─ Sub-Location (Zona)
      └─ Items
```

Estructura Plana en SIESA

```
Bodega
  └─ Items (sin sub-ubicaciones)
```

Mapeo

- Kong mantiene jerarquía detallada de ubicaciones
- SIESA solo recibe saldos totales por bodega principal
- La trazabilidad detallada se mantiene solo en Kong

Consultas Relacionales Comunes

Consulta 1: Obtener Inventario Total por SKU

javascript

```
// Kong: Agrupar items por SKU y ubicación
SELECT
  sku.external_id,
  location.external_id,
  COUNT(item.id) as quantity
FROM items
JOIN skus ON items.sku_id = skus.id
JOIN locations ON items.location_id = locations.id
WHERE items.status = 'AVAILABLE'
GROUP BY sku.id, location.id

// SIESA: Consultar saldo directo
GET /api/consultas/saldos?item=PROD-001&bodega=BOD01
```

Consulta 2: Movimientos Pendientes de Sincronizar

javascript

```
// Kong: Moves cerrados sin documento SIESA
SELECT *
FROM moves
WHERE status = 'CLOSED'
  AND reference NOT LIKE 'SIESA-%'
  AND closed_at > '2025-10-01'
```

Consulta 3: Diferencias de Inventario

javascript

```
// Comparar saldos Kong vs SIESA
const diferencias = [];

for (const sku of skus) {
  const saldoKong = await getSaldoKong(sku.external_id, 'BOD01');
  const saldoSIESA = await getSaldoSIESA(sku.external_id, 'BOD01');

  if (saldoKong !== saldoSIESA) {
    diferencias.push({
      sku: sku.external_id,
      kong: saldoKong,
      siesa: saldoSIESA,
      diferencia: saldoKong - saldoSIESA
    });
  }
}
```

Consideraciones de Integridad Referencial

Validaciones Pre-envío a SIESA

javascript

```
async function validarIntegridadReferencial(documento) {
  const errores = [];

  // 1. Validar que items existan en SIESA
  for (const mov of documento.Movimientos) {
    const itemExiste = await verificarItemSIESA(mov.f470_id_item);
    if (!itemExiste) {
      errores.push(`Item ${mov.f470_id_item} no existe en SIESA`);
    }
  }

  // 2. Validar que bodegas existan
  const bodegaEntrada = documento.Documentos[0].f450_id_bodega_entrada;
  const bodegaSalida = documento.Documentos[0].f450_id_bodega_salida;

  if (bodegaEntrada) {
    const existe = await verificarBodegaSIESA(bodegaEntrada);
    if (!existe) {
      errores.push(`Bodega ${bodegaEntrada} no existe en SIESA`);
    }
  }

  if (bodegaSalida) {
    const existe = await verificarBodegaSIESA(bodegaSalida);
    if (!existe) {
      errores.push(`Bodega ${bodegaSalida} no existe en SIESA`);
    }
  }

  // 3. Validar tercero si aplica
  const tercero = documento.Documentos[0].f350_id_tercero;
  if (tercero) {
    const existe = await verificarTerceroSIESA(tercero);
```

```
    if (!existe) {
      errores.push(`Tercero ${tercero} no existe en SIESA`);
    }
  }

  return errores;
}
```

Estrategia de Resolución de Conflictos

javascript

```
// Si un item no existe en SIESA pero existe en Kong
async function resolverConflictoItem(itemKong) {
  // Opción 1: Omitir la línea y registrar error
  console.error(`Item ${itemKong} no sincronizado, omitiendo línea`);

  // Opción 2: Intentar sincronizar antes de enviar
  const productoSIESA = await consultarItemSIESA(itemKong);
  if (productoSIESA) {
    await sincronizarProductoAKong(productoSIESA);
  }

  // Opción 3: Marcar para revisión manual
  await registrarConflicto({
    tipo: 'item_no_encontrado',
    item: itemKong,
    timestamp: new Date()
  });
}
```

Resumen de Relaciones Críticas

Relación	Tipo	Dirección	Frecuencia	Criticidad
Productos	1:1	SIESA → Kong	Horaria	Alta
Clientes	1:1	SIESA → Kong	Diaria	Media
Bodegas	1:1	SIESA → Kong	Ante cambios	Alta
Órdenes Compra	1:1	SIESA → Kong	Tiempo real	Media
Órdenes Venta	1:1	SIESA → Kong	Tiempo real	Alta
Movimientos	N:1	Kong → SIESA	Tiempo real	**Crítica**
Ajustes	N:M	Kong → SIESA	Post-conteo	Alta

Nota: La relación de Movimientos Kong → SIESA es la más crítica ya que impacta directamente los saldos contables.