

Verification of Business Cases Against Data Model Design

1. Introduction

This report aims to verify whether the provided business cases are supported by the existing data model design. The verification focuses on the alignment between business requirements, table structures, relationships (foreign keys, many-to-many associations), and constraint definitions. The analysis confirms that the model effectively accommodates the core needs of each case.

2. Case Verification & Model Support Analysis

2.1 Account - BillingAccount (n..n) Business Cases

- **Business Needs:** Walmart Columbus requires 3 PO boxes for FSA, Life, and A&H invoices (distinguished by business type and employee type); Keith's Garage uses a single address for all business, and other companies entrust Keith's Garage with paperwork.
- **Model Support:**
 - The many-to-many relationship between Account and BillingAccount is implemented via the junction table Account_BillingAccount, which supports multiple billing accounts linked to a single account.
 - BillingAccount includes POBox (stores address information), BusinessType (constrained to FSA/Life/A&H, matching invoice types), and EmployeeType (constrained to salaried/hourly/ALL, supporting employee type differentiation).
 - Keith's Garage can use one BillingAccount record (with a single POBox) to cover all business needs, and the junction table allows other companies (Account) to associate with this billing account, fulfilling the paperwork entrustment scenario.

2.2 Account - Account (n..n) Business Cases

- **Business Needs:** The Archdiocese of Orlando has St. Jude's Church and St. Mary's Church as members; St. Jude's Church designates the Archdiocese as GroupMaster and Pepsi Co. as Flex Master.
- **Model Support:**
 - The many-to-many relationship between Account and Account is realized through Account_Account, with RelationType constrained to GroupMaster/Flex Master/Member.
 - St. Jude's Church (MemberAccountID) can associate with the Archdiocese (MainAccountID, RelationType=GroupMaster) and Pepsi Co. (MainAccountID, RelationType=Flex Master) via separate records in

`Account_Account`, fully supporting the hierarchical and multi-master account requirements.

2.3 Account - AccountAdmin (n..n) Business Cases

- Business Needs: Walmart Columbus uses 3 administrators specialized in FSA, Life, and A&H; an administrator can service multiple accounts.
- Model Support:
 - The many-to-many association between `Account` and `AccountAdmin` is established via `Account_AccountAdmin`.
 - `AccountAdmin` has an `Expertise` field (constrained to FSA/Life/A&H), enabling Walmart to associate with 3 administrators of different specializations.
 - The junction table allows a single `AccountAdmin` to link to multiple `Account` records, supporting cross-account service scenarios.

2.4 Account - AccountMember (1..n) Business Cases

- Business Needs: Keith's Garage has multiple employees; Shannon has work history across employers (retaining policies) and moonlights; 13-month record retention for employment history.
- Model Support:
 - `AccountMember` includes `AccountId` (foreign key to `Account`), establishing a one-to-many relationship (one account has multiple members).
 - Fields `IsCurrent` (identifies current employment), `StartDate`, and `EndDate` support tracking of employment history (including 13-month retention).
 - Shannon's moonlighting scenario is accommodated by multiple `AccountMember` records (linking to different `Account` IDs for Keith's Garage and Gary's Trucking Co.), with date fields distinguishing employment periods.

2.5 AccountAlias Business Cases

- Business Needs: Consolidation of GroupMaster/ Flex data into `Account`; retention of original records and duplicate flags for manual review.
- Model Support:
 - `AccountAlias` is designed to store original data before consolidation, with `OriginalDataType` (recording GroupMaster/Flex sources) and `DuplicateFlag` (identifying duplicates).

- A single Account (post-consolidation) can link to multiple AccountAlias records via AccountId, supporting Gene's Auto Parts' 3 original records and the duplicate GroupMaster entry scenario.

2.6 Account CompanyCode Logical Primary Key Business Cases

- Business Needs: Legitimate distinction between same-named companies in different regions; invalidation of same-name/same-region duplicates.
- Model Support:
 - While AccountId is the physical primary key, the CompanyCode field is designated as a logical business key to distinguish regional attributes.
 - CompanyCode can be structured to include region identifiers (e.g., "XYZ-C Columbus" vs. "XYZ-NY New York"), enabling legitimate differentiation of same-named companies in different regions and preventing same-region duplicates through business rule enforcement.

2.7 Account - ManagerContract (n..n) Business Cases

- Business Needs: An Associate services multiple Accounts via a specific ManagerContract; an Account has multiple Associate roles (Original Servicing, Broker, etc.).
- Model Support:
 - The many-to-many relationship between Account and ManagerContract is implemented via Account_ManagerContract, with RoleType constrained to Original Servicing/Servicing/Broker.
 - ManagerContract links to Associate via AssociateId, enabling an Associate to service multiple Accounts through different ManagerContract records, and an Account to associate with multiple Associates via distinct RoleType entries.

2.8 Associate - ManagerContract (1..n) Business Cases

- Business Needs: An Associate holds multiple Writing Numbers and SitCodes (formal reporting chain + side deals).
- Model Support:
 - ManagerContract includes AssociateId (foreign key to Associate), establishing a one-to-many relationship (one Associate has multiple ManagerContracts).
 - SitCode and SitCodeType (Formal/Informal) support Keith's scenario: SitCode0 (Formal, linking to DSC/RSC) and SitCodeA (Informal, side deal with Jim's upline chain).

- WritingNumber in ManagerContract accommodates multiple authorization numbers per Associate, aligned with state licensing requirements.

2.9 Associate Commissions vs. Production Credit Business Cases

- Business Needs: Dual accounting of Contract Premiums (Commissions for payments, Production Credit for contests).
- Model Support:
 - ContractPremium includes CalculationType (constrained to Commission/Production Credit), directly supporting the two accounting methods.
 - The table links to ContractBenefit and Contract, ensuring premiums are traced to specific policies, enabling accurate commission disbursement and contest scoring.

2.10 Associate - Associate (n..n) Business Cases

- Business Needs: Associate relationships including Broker, Recruiter, Upline, etc.
- Model Support:
 - The many-to-many relationship between Associate and Associate is realized via Associate_Associate, with RelationType constrained to Broker/Recruiter/Upline/DSC/RSC, fully covering all required relationship types.

2.11 Contract - ContractPremium (1..n..1) Business Cases

- Business Needs: A Contract has multiple benefits, each with multiple premiums (policy updates for marriage, children).
- Model Support:
 - Contract links to ContractBenefit (one-to-many, via ContractId), and ContractBenefit links to ContractPremium (one-to-many, via ContractBenefitId), forming a 1→n→n hierarchy.
 - This structure supports Keith's policy updates: one Contract (original policy) with multiple ContractBenefit records (spouse, children) and corresponding ContractPremium entries for each benefit, tracked by Period.

2.12 Customer - Customer (n..n) Business Cases

- Business Needs: Family, Trust, PolicyOwner, and Beneficiary relationships (Daddy Warbucks, son, trust).
- Model Support:

- Customer_Customer junction table with RelationType (Family/Trust/PolicyOwner/Beneficiary) supports all required relationships.
- Daddy Warbucks (MainCustomerId) can link to his son (RelatedCustomerId, RelationType=Family) and the trust (RelatedCustomerId, RelationType=Trust), fulfilling the policy structure scenario.

2.13 Customer - Contract (n..n) Business Cases

- Business Needs: Policy Owner (Daddy Warbucks) and Payer (trust) distinction; personal policies for the owner.
- Model Support:
 - Contract includes PolicyOwnerID and PayerID (both foreign keys to Customer), enabling separate designation of policy owner and payer (Daddy as owner, trust as payer).
 - Daddy's personal life policy is supported by a Contract record where PolicyOwnerID and PayerID both reference his CustomerId.

2.14 Customer - ContractBenefit (n..n) Business Cases

- Business Needs: Multiple beneficiaries for a policy (son and family); accident insurance coverage.
- Model Support:
 - The many-to-many relationship between Customer and ContractBenefit is established via ContractBenefit_Customer, allowing multiple beneficiaries (son, family members) to link to a single ContractBenefit.
 - Separate ContractBenefit records (health insurance, accident insurance) can be associated with the son's CustomerId, supporting comprehensive coverage scenarios.

2.15 Customer - Account (n..n) Business Cases

- Business Needs: Employees linked to company accounts; moonlighting across multiple companies.
- Model Support:
 - Account_Customer junction table connects Account (companies) and Customer (employees), with IsMoonlighting indicating moonlighting status.
 - Keith's association with Dana's Dry Cleaning and Scott's Garage is supported by two Account_Customer records, with IsMoonlighting set to 1 for the secondary employment.

2.16 Customer - Associate (n..n) Business Cases

- Business Needs: Commission beneficiary designations (Walt's wife/son, Dave's bequest to Walt's son).
- Model Support:
 - Customer_Associate junction table with RelationType (Commission Beneficiary) supports associating Associates (Walt, Dave) with their designated commission beneficiaries (Walt's wife, Walt's son).
 - Multiple records in Customer_Associate allow Walt's son to be linked to both Walt and Dave as a commission beneficiary.

3. Conclusion

The existing data model design fully supports all presented business cases. The table structures, many-to-many junction tables, constraint definitions (CHECK constraints for enumeration values), and foreign key relationships are aligned with business requirements, ensuring accurate capture of entities, attributes, and their associations. The model demonstrates strong flexibility and scalability to accommodate diverse business scenarios, including hierarchical account relationships, multi-dimensional policy tracking, and complex associate-customer interactions.