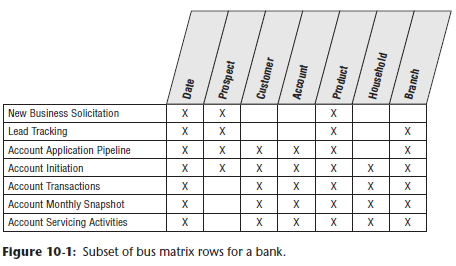
# Kimball Data Warehouse Toolkit

## Ch 11 – Financial Services

* The financial services industry encompasses a wide variety of businesses, including credit card companies, brokerage firms, and mortgage providers
* We’ll primarily focus on the retail bank since most readers have some degree of personal familiarity with this type of financial institution
* A full-service bank offers a breadth of products, including checking accounts, savings accounts, mortgage loans, personal loans, credit cards, and safe deposit boxes
* We begin w/ a very simplistic schema + then explore several schema extensions, including the handling of the bank’s broad portfolio of heterogeneous products that vary significantly by line of business
* NOTE: Industry-focused chapters like this one are not intended to provide full-scale industry solutions
* Although various dimensional modeling techniques are discussed in the context of a given industry, the techniques are certainly applicable to other businesses
* If you don’t work in financial services, you still need to read this chapter
* If you do work in financial services, remember these schemas should not be viewed as *complete*
* **Concepts:**
* Bus matrix snippet for a bank
* **Dimension triage** to avoid the “too few dimensions” trap
* Household dimensions
* **Bridge tables** to associate multiple customers with an account, along with **weighting factors**
* Multiple **mini-dimensions** in a single fact table
* **Dynamic value banding** of facts for reporting
* Handling heterogeneous products across lines of business, each with unique metrics and/or dimension attributes, as supertype and subtype schemas
* **Hot swappable dimensions**

### Banking Case Study and Bus Matrix

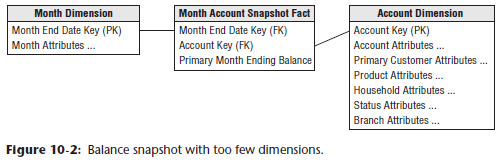
* The **bank’s initial goal** is to **better analyze the bank’s accounts**
* Business users want the ability to slice + dice individual accounts, as well as the residential household groupings to which they belong
* **One of the bank’s major objectives is to market more effectively by offering additional products to households that already have one or more accounts with the bank**
* The figure illustrates a *portion* of a bank’s bus matrix.



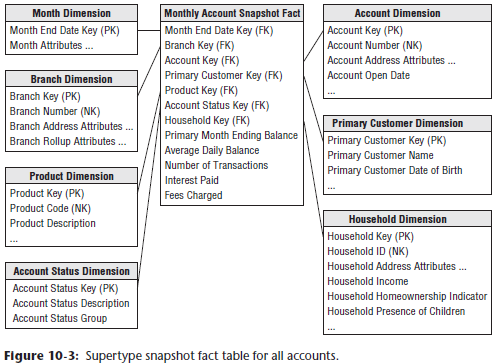
* After conducting interviews with managers + analysts around the bank, the following set of requirements were developed:
* Business users want to see 5 years of historical monthly snapshot data on every account
* Every account has a **primary balance**, + the business wants to group different types of accounts in the same analyses + compare primary balances
* Every ***type* of** **account (known as *products* within the bank)** has a set of custom dimension attributes and numeric facts that tend to be quite different from product to product
* Every account is deemed to belong to a single household
* There is a surprising amount of volatility in the account/household relationships due to changes in marital status and other life stage factors
* In addition to the household identification, users are interested in demographic information both as it pertains to individual customers and households
* In addition, the bank captures + stores behavior scores relating to the activity or characteristics of each account and household

### Dimension Triage to Avoid Too Few Dimensions

* Based on the previous business requirements, the grain and dimensionality of the initial model begin to emerge
* **Can start with a fact table that records the primary balances of every account at the end of each month**
* Clearly, the **grain of this fact table is 1 row for each account each month**
* Based on that grain declaration, you can initially envision a design with only 2 dimensions: month and account
* These 2 FKs form the fact table PK, as shown below



* A data-centric designer might argue that all the other description information, such as household, branch, and product characteristics should be embedded as descriptive attributes of the account dimension because each account has only one household, branch, + product associated with it
* **Although the above schema *accurately* represents the many-to-one and many-to-many relationships in the snapshot data, it does NOT adequately reflect the natural business dimensions**
* **Rather than collapsing everything into the huge account dimension table, additional analytic dimensions such as product and branch mirror the instinctive way users think about their business**
* These **supplemental dimensions provide much smaller points of entry to the fact table** +, thus, they **address both the performance + usability objectives of a dimensional model**
* Finally, **given a big bank may have millions of accounts, you should worry about type 2 SCD effects potentially causing this huge dimension to mushroom into something unmanageable**
* The product + branch attributes are convenient groups of attributes to remove from the account dimension to cut down on the row growth caused by type 2 SCD change tracking
* In “Mini-Dimensions Revisited,” the changing demographics + behavioral attributes will be squeezed out of the account dimension for the same reasons
* The **product + branch dimensions are 2 separate dimensions, as there is a many-to-many relationship between products and branches**
* **They both change slowly, but on different rhythms**
* **Most important, *business users* think of them as distinct dimensions of the banking business**
* **In general, most dimensional models end up with between 5-20 dimensions**
* If at or below the low end of this range, be suspicious that dimensions may have been inadvertently left out of the design
* In this case, carefully consider whether any of the following kinds of dimensions are appropriate supplements to your initial dimensional model:
* **Causal dimensions**, such as promotion, contract, deal, store condition, or even weather
* These dimensions, as discussed in Chapter 3: Retail Sales**, provide additional insight into the cause of an event.**
* ***Multiple* date dimensions**, *especially when the fact table is an accumulating snapshot*
* Refer to Chapter 4: Inventory for sample fact tables with multiple date stamps
* **Degenerate dimensions that identify operational transaction control numbers**, such as an order, an invoice, a bill of lading, or a ticket, as initially illustrated in Chapter 3
* **Role-playing dimensions**, such as when a single transaction has several business entities associated with it, each represented by a separate dimension
* In Chapter 6: Order Management, we described role playing to handle multiple dates
* **Status dimensions** that identify the current status of a transaction or monthly snapshot within some larger context, such as an account status.
* **An audit dimension**, as discussed in Chapter 6, to track data lineage + quality
* **Junk dimensions** of correlated indicators and flags, as described in Chapter 6
* **These dimensions can typically be added gracefully to a design, even after the DW/BI system has gone into production, because they do not change the grain of the fact table**
* The addition of these dimensions usually does NOT alter the existing dimension keys or measured facts in the fact table
* All existing applications should continue to run without change
* **NOTE: Any descriptive attribute that is single-valued in the presence of the measurements in the fact table is a good candidate to be added to an existing dimension or to be its own dimension**
* Based on further study of the bank’s requirements, you can ultimately choose the following dimensions for the initial schema: month end date, branch, account, primary customer, product, account status, and household
* As illustrated below, **at the intersection of these 7 dimensions, you take a monthly snapshot + record the primary balance + any other metrics that make sense across all products, such as transaction count, interest paid, + fees charged**



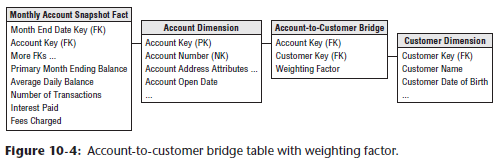
* **Remember: account balances are just like inventory balances, in that they are NOT additive across ANY measure of time**
* **Instead, you must *average* the account balances by dividing the balance sum by the number of time periods**
* **NOTE:** In this chapter we use the basic object-oriented terms **supertype**and **subtype**to **refer respectively to the single fact table covering all possible account types, as well as the multiple fact tables containing specific details of each individual account type**
* **In past writings these have been called *core* and *custom* fact tables**, but it is time to change to the more familiar and accepted terminology
* The **product dimension** consists of a **simple hierarchy that describes all the bank’s products**, including the name of the product, type, + category
* The need to construct a generic product categorization in the bank is the same need that causes grocery stores to construct a generic merchandise hierarchy
* The main difference between the bank and grocery store examples is that the bank *also* develops a large number of *subtype* product attributes for each product type
* See the “Supertype and Subtype Schemas for Heterogeneous Products” section at the end of the chapter
* The **branch dimension** is **similar to the facility dimensions** we discussed earlier in this book, such as the retail store or distribution center warehouse.
* The **account status dimension** is a useful dimension to **record the condition of the account at the end of each month**
* The status records whether the account is active or inactive, or whether a status change occurred during the month, such as a new account opening or account closure
* Rather than whipsawing the large account dimension, or merely embedding a cryptic status code or abbreviation directly in the fact table, we treat status as a full-fledged dimension with descriptive status decodes, groupings, and status reason descriptions, as appropriate
* In many ways, you could consider the account status dimension to be another example of a **mini-dimension**, as introduced in Chapter 5: Procurement

#### Household Dimension

* Rather than focusing solely on the bank’s accounts, business users also want the ability to analyze the bank’s relationship with an **entire economic unit**, referred to as a **household**
* They’re interested in understanding the overall profile of a household, the magnitude of the existing relationship with the household, + what additional products should be sold to the household
* They also want to capture key demographics regarding the household, such as household income, whether they own or rent, whether they are retirees, + whether they have children
* **These demographic attributes change over time +, as you might suspect, the users want to track the changes**
* If the bank focuses on accounts for commercial entities, rather than consumers, similar requirements to identify and link corporate “households” are common
* **From the bank’s perspective, a household may be comprised of several accounts + individual account holders**
* Ex: Consider John and Mary Smith as a single household
* John has a checking account, whereas Mary has a savings account
* In addition, they have a joint checking account, credit card, + mortgage with the bank
* All 5 of these accounts are considered to be a part of the same Smith household, despite the fact that minor inconsistencies may exist in the operational name + address information
* **The process of relating individual accounts to households (or the commercial business equivalent) is not to be taken lightly**
* **“Householding” requires the development of business rules + algorithms to assign accounts to households**
* There are **specialized products + services to do the matching necessary to determine household assignments**
* **Very common for a large financial services organization to invest significant resources in specialized capabilities to support its householding needs**
* **The decision to treat account and household as separate dimensions is somewhat a matter of the designer’s prerogative**
* **Even though they are intuitively correlated, you decide to treat them separately because of the size of the account dimension + the volatility of the account constituents within a household dimension**, as mentioned earlier
* In a large bank, the account dimension is huge, with easily over 10M rows that group into several million households
* **The household dimension provides a somewhat smaller point of entry into the fact table, without traversing a 10M-row account dimension table**
* Also, **given the changing nature of the relationship between accounts + households, you elect to use the fact table to capture the relationship, rather than merely including the household attributes on each account dimension row**
* *This way, you* ***avoid using the type 2 SCD technique w/ a 10-million row account dimension***

#### Multivalued Dimensions and Weighting Factors

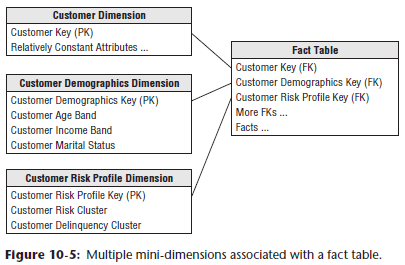
* As you just saw in the John and Mary Smith example, **an “account” can have 1, 2, or more individual account holders, or customers, associated with it**
* Obviously, **customer cannot be included as an *account* attribute (beyond the designation of a primary customer/account holder), as doing so violates the granularity of the dimension table, because more than 1 individual can be associated with an account**
* Likewise, you **cannot include a customer as an additional dimension in the fact table, as doing so violates the granularity of the fact table (1 row per account per month), *again* because more than one individual can be associated with any given account**
* This is another classic example of a **multivalued dimension**
* **To link an individual *customer* dimension to an *account*-grained fact table requires the use of an account-to-customer bridge table**, as shown below



* **At a minimum, the PK of the bridge table consists of the surrogate account + customer keys.**
* The timestamping of bridge table rows, as discussed in Chapter 7: Accounting, for time-variant relationships is also applicable in this scenario
* **If an account has 2 account holders, the associated bridge table has 2 rows**
* You **assign a numerical weighting factor to each account holder such that the sum of all the weighting factors is exactly 1.00**
* The **weighting factors are used to allocate any of the numeric additive facts across individual account holders**
* In this way, you **can add up all numeric facts *by individual holder*, + the grand total will be the correct grand total amount**
* This kind of report is a **correctly-weighted report**
* **Weighting factors = simply a way to allocate numeric additive facts across account holders**
* Some would suggest changing the grain of the fact table to be account snapshot by account holder
* In this case, you’d **take the weighting factors + physically multiply them against the original numeric facts**
* This is **rarely done for 3 reasons**
* **1) The size of the fact table would be multiplied by the average number of account holders**
* **2) Some fact tables have more than 1 multivalued dimension, the number of rows would get out of hand in this situation, + you’d start to question the physical significance of an individual row**
* **3) You may want to see the unallocated numbers, + it is hard to reconstruct these if the allocations have been combined physically with the numeric facts.**
* If you **choose *not* to apply the weighting factors in a given query**, you **can still summarize the account snapshots by individual account holder, but in this case, you get what is called an** **impact report**
* A question such as, “What is the total balance of all individuals with a specific demographic profile?” would be an example of an impact report
* **Business users understand impact analyses may result in overcounting because the facts are associated with *both* account holders**
* In the above schema, a **SQL view could be defined, combining the fact table + the account-to-customer bridge table so these 2 tables, when combined, would appear to BI tools as a standard fact table with a normal customer FK**
* And***two* views could be defined, one using weighting factors, one *not* using weighting factors**
* **NOTE: An open-ended, many-valued attribute can be associated with a dimension row by using a bridge table to associate the many-valued attributes with the dimension**
* In some financial services companies, the **individual customer is identified + associated with each transaction**
* Ex: Credit card companies often issue unique card numbers to each cardholder
* John and Mary Smith may have a joint credit card account, but the numbers on their respective pieces of plastic are unique
* **In this case, there is *no* need for an account-to-customer bridge table because the atomic transaction facts are at the discrete customer grain, and account + customer would both be FKs in this fact table**
* ***However*, the bridge table would be required to analyze metrics that are naturally captured at the account level, such as the credit card billing data**

#### Mini-Dimensions Revisited

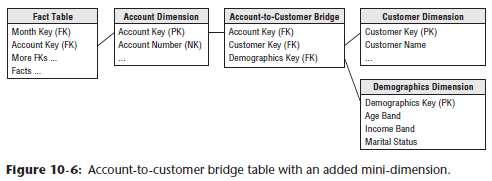
* Similar to the discussion of the customer dimension in Chapter 8: CRM, **there are a wide variety of attributes describing the bank’s accounts, customers, + households**, including monthly credit bureau attributes, external demographic data, + calculated scores to identify their behavior, retention, profitability, + delinquency characteristics
* **Financial services organizations are typically interested in understanding + responding to changes in these attributes over time**
* As discussed earlier, **it’s unreasonable to rely on type 2 SCD technique to track changes in the account dimension, given the dimension row count + attribute volatility, such as the monthly update of credit bureau attributes**
* Instead, **can break off the browse-able + changeable attributes into multiple mini-dimensions**, such as credit bureau + demographics mini-dimensions, whose keys are included in the fact table, as illustrated below



* **The type 4 SCD mini-dimensions enable you to slice + dice the fact table, while readily tracking attribute changes over time, even though they may be updated at different frequencies**
* ***Although mini-dimensions are extremely powerful, be careful to avoid overusing the technique***
* **Account-oriented financial services are a good environment for using mini-dimensions because the primary fact table is a very long-running periodic snapshot**
* Thus, every month a fact table row is *guaranteed* to exist for every account, providing a home for all the associated FKs
* You can always see the account together with all the mini-dimensions for any month
* **NOTE: Mini-dimensions should consist of *correlated* clumps of attributes, + each attribute *shouldn’t* be its own mini-dimension or you end up with too many dimensions in the fact table**
* As described in Chapter 4, **one of the compromises associated with mini-dimensions is the need to band attribute values to maintain reasonable mini-dimension row counts**
* **Rather than storing extremely discrete income amounts, such as $31,257.98, store income *ranges***, such as $30,000 to $34,999 in the mini-dimension
* Similarly, the profitability scores may range from 1 through 1200, which you band into fixed ranges such as less than or equal to 100, 101 to 150, and 151 to 200, in the mini-dimension
* **Most organizations find these banded attribute values support their routine analytic requirements, however there are 2 situations in which banded values may be inadequate**
* **1) Data mining analysis often requires discrete values rather than fixed bands to be effective**
* **2) A limited number of power analysts may want to analyze the discrete values to determine if the bands are appropriate**
* In this case, you **still maintain the banded value mini-dimension attributes** to support consistent day-to-day analytic reporting **but *also* store the key discrete numeric values as facts in the fact table**
* Ex: If each account’s profitability score were recalculated each month, you’d assign the appropriate profitability range mini-dimension for that score each month
* In addition, you’d capture the *discrete* profitability score as a fact in the monthly account snapshot fact table
* Finally, **if needed, the current profitability range or score could be included in the *account* dimension, where any changes are handled by deliberately overwriting the type 1 SCD attribute**
* **Each of these data elements should be *uniquely* labeled so that they are distinguishable**
* **Designers must always carefully balance the incremental value of including such somewhat redundant facts and attributes versus the cost in terms of additional complexity for both the ETL processing + BI presentation**

#### Adding a Mini-Dimension to a Bridge Table

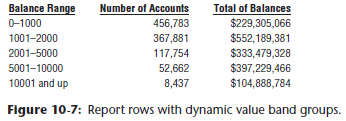
* In the bank account example, the **account-to-customer bridge table can get very large**
* **W/ 20M accounts + 25M customers, the bridge table can grow to hundreds of millions of rows after a few years if both the account dimension + the customer dimension are type 2 SCDs (where you track history by issuing new rows with new keys)**
* Now the experienced dimensional modeler asks, **“What happens when my customer dimension turns out to be a rapidly-changing monster dimension?”**
* **Could happen when rapidly changing demographics + status attributes are added to the customer dimension, forcing numerous type 2 additions to the customer dimension**
* Now the 25M row customer dimension threatens to become several hundred million rows
* **Standard response to a rapidly changing monster dimension = split off the rapidly-changing demographics + status attributes into a type 4 mini-dimension, often called a demographics dimension**
* This **works great when this dimension attaches *directly* to the fact table along with a customer dimension, because it stabilizes the large customer dimension and keeps it from growing every time a demographics or status attribute changes**
* But **can you get this same advantage when the customer dimension is attached to a *bridge* table**, as in the bank account example?
* **The solution = Add a FK reference in the bridge table to the demographics dimension**, as shown below



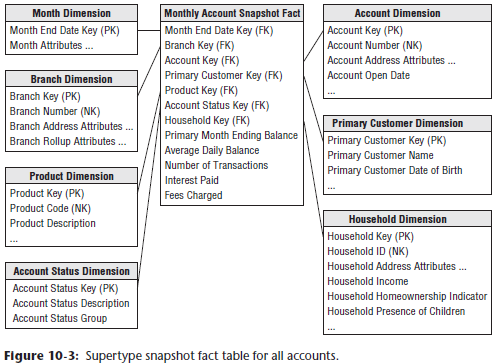
* The way to visualize **this bridge table** is that it **links every account to its associated customers + their demographics**
* **Key for this bridge table now consists of the account key, customer key, + demographics key**
* **Depending on how frequently new demographics are assigned to each customer, the bridge table will perhaps grow significantly**
* In the above design, because the grain of the root bank account fact table is month by account, the bridge table should be limited to changes recorded only at month ends
* This takes some of the change tracking pressure off the bridge table

#### Dynamic Value Banding of Facts

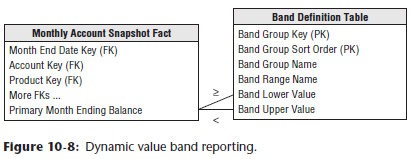
* **Suppose business users want the ability to perform value band reportingon a standard numeric fact, such as account balance, but are NOT willing to live w/ pre-defined bands in a dimension table**
* They may want to create a report based on the account balance snapshot, as shown below



* Using the schema from before (below), it is difficult to create this report directly from the fact table



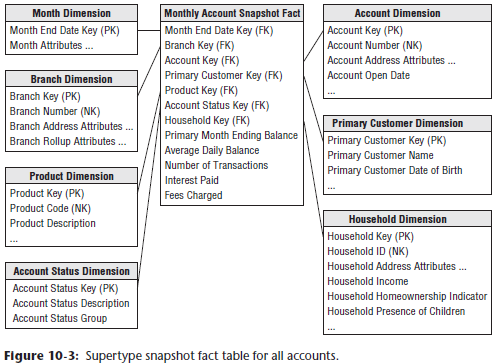
* **SQL has no generalization of the GROUP BY clause that clumps additive values into ranges**
* To further complicate matters, the **ranges are of unequal size + have textual** **names**, such as “10001 and up.”
* Also, **users typically need the flexibility to redefine the bands at query time with different boundaries or levels of precision.**
* The **new schema design shown below enables on-the-fly value band reporting**



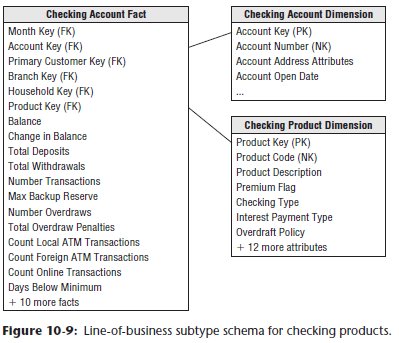
* The **band definition table** **can contain as many sets of different reporting bands as desired**
* The name of a particular group of bands is stored in the band group column
* The band definition table is joined to the balance fact table using a pair of less-than + greater-than JOINs
* The **report uses the band range *name* as the row header + sorts the report on the sort order attribute**
* **Controlling the performance of this query can be a challenge**
* **A value band query is by definition very lightly constrained**
* The example report above needed to scan the balances of > 1M accounts
* Perhaps only the month dimension was constrained to the current month
* Furthermore, the funny JOINs to the value banding table are *NOT* the basis of a nice restricting constraint, because they are grouping the 1M balances
* **In this situation, you may need to place an index directly on the balance fact**
* **The performance of a query that constrains/groups on the value of a fact-like balance will be improved enormously if the DBMS can efficiently sort + compress the individual fact**
* This approach was pioneered by the Sybase IQ columnar database product in the early 1990s + is now becoming a standard indexing option on several of the competing columnar DBMSs.

### Supertype and Subtype Schemas for Heterogenous Products

* In many financial service businesses, a **dilemma arises because of the heterogeneous nature of the products or services offered by the institution**
* As mentioned in the intro, **a typical retail bank offers a *myriad* of products, from checking accounts to credit cards, to the *same* customers**
* Although every account at the bank has a primary balance + interest amount associated with it, **each product type has many special attributes + measured facts *not* shared by the other products**
* Ex: Checking accounts have minimum balances, overdraft limits, service charges, + other measures relating to online banking
* Time deposits such as certificates of deposit have few attribute overlaps with checking, but have maturity dates, compounding frequencies, + current interest rate
* **Business users typically require 2 different perspectives difficult to present in a single fact table**
* **1) The global view, including the ability to slice + dice all accounts simultaneously, regardless of their product type**
* This global view is needed to plan appropriate CRM cross-sell and up-sell strategies against the aggregate customer/household base spanning all possible products
* In this situation, you’d the **single supertype fact table** (below) **that crosses all the lines of business to provide insight into the complete account portfolio**



* Note, however, that **the supertype fact table can present only a *limited* number of facts that make sense for virtually *every* line of business**
* **Cannot accommodate incompatible facts in a supertype fact table, because there may be several hundred of these facts when all the possible account types are considered**
* Similarly, the **supertype product dimension must be restricted to the subset of common product attributes**
* **2) The line-of-business view, which that focuses on the in-depth details of *one* business, such as checking**
* There’s a long list of **special facts + attributes that make sense only for the checking business**
* These **special facts *cannot* be included in the supertype fact table**
* If you did this for each line of business in a retail bank, you’d **end up with hundreds of special facts, most of which would have NULL values in any specific row**
* Likewise, **if you attempt to include line-of-business attributes in the account or product dimension tables, these tables would have hundreds of special attributes, almost all of which would be empty for any given row**
* The resulting tables would resemble Swiss cheese, littered with data holes
* **Solution to this dilemma** **for this checking department** example = **create a subtype schema for the checking line of business that’s limited to *just* checking accounts**, like below



* Now, **both the subtype checking fact table and corresponding checking account dimension are widened to describe all the specific facts and attributes that make sense only for checking products**
* These **subtype schemas must *also* contain the supertype facts + attributes to avoid JOIN-ing tables from the supertype + subtype schemas for the complete set of facts and attributes**
* **Can also build separate subtype fact + account tables for the *other* lines of business, to support their in-depth analysis requirements**
* Although creating account-specific schemas sounds complex, only the DBA sees all tables at once
* From the business users’ perspective, either it’s a cross-product analysis that relies on the single supertype fact table + its attendant supertype account table, or the analysis focuses on a particular account type + only one of the subtype line of business schemas is utilized
* **In general, it makes less sense to combine data from more than 1 subtype schema, because by definition, the accounts’ facts and attributes are disjointed (or nearly so)**
* The **keys of the subtype account dimensions are the same keys used in the supertype account dimension, which contains all possible account keys**
* Ex: If the bank offers a “$500 minimum balance with no per check charge” checking account, this account would be identified by the same surrogate key in both the supertype and subtype checking account dimensions
* **Each subtype account dimension is a shrunken conformed dimension with a subset of rows from the supertype account dimension table, + each subtype account dimension contains attributes specific to a particular account type**
* This **supertype/subtype design technique applies to any business that offers widely varied products through multiple lines of business**
* Ex: A tech company that sells hardware, software, + services, you can imagine building supertype sales fact + product dimension tables to deliver the global customer perspective
* Supertype tables would include all facts + dimension attributes common across lines of business
* Supertype tables would then be supplemented w/ schemas that do a deep dive into subtype facts + attributes that vary by business
* Again, a specific product would be assigned the same surrogate product key in both the supertype and subtype product dimensions.
* **NOTE: A family of supertype + subtype fact tables are needed when a business has heterogeneous products that have naturally different facts + descriptors, but a single customer base that demands an integrated view**
* If the lines of business in a retail bank are physically separated so each has its own location, the subtype fact + dimension tables will likely not reside in the same space as the supertype fact + dimension tables
* In this case, the data in the supertype fact table would be duplicated exactly once to implement all the subtype tables.
* **Remember that the subtype tables provide a disjointed partitioning of the accounts, so there is no overlap between the subtype schemas**

#### Supertype and Subtype Products with Common Facts

* The **supertype + subtype product technique** just discussed is **appropriate for fact tables where a single logical row contains many product-specific facts**
* ***On******the other hand*, metrics captured by some business processes**, such as the bank’s new account solicitations, **may *not* vary by line of business**
* **In this case, you do NOT need line-of-business fact tables, + one supertype fact table suffices**
* However, you **still can have a rich set of heterogeneous products with diverse attributes**
* In this case, you’d generate the complete portfolio of subtype account dimension tables, + use them as appropriate, depending on the nature of the application.
* In a cross-product analysis, the supertype account dimension table would be used because it can span any group of accounts
* In a single account type analysis, you could optionally use the subtype account dimension table instead of the supertype dimension if you wanted to take advantage of the subtype attributes specific to that account type

### Hot Swappable Dimensions

* A brokerage house may have many clients who track the stock market, + all of them access the same fact table of daily high-low-close stock prices
* But each client has a confidential set of attributes describing each stock
* The brokerage house can support this multi-client situation by **having a separate copy of the stock dimension for each client**, which is **JOIN-ed to the single fact table at query time**
* We call these **hot swappable dimensions**
* **To implement hot swappable dimensions in a *relational* environment, referential integrity constraints between the fact table + the various stock dimension tables probably must be turned *OFF* to allow the switches to occur on an individual query basis**