SEAD ENERGY EFFICIENCY DATA ACCESS PROJECT: FINAL REPORT

October, 2013

SEAD Energy Efficiency Data Access Project: Final Report

Alex Katzman
Michael McNeil
Brian Gerke

October 2013

Table of Contents

Introduction

The Internet retailing revolution has had profound effects not only on the way we buy products, but also in the availability of market data and ease of collecting it. This also includes household appliances and their efficiency characteristics. Development of a database framework is the first step in fully exploiting the newfound abundance of publically available appliance efficiency data and provides a technical foundation on which to build data collection, display and analysis tools for a variety of objectives and audiences.

There are two main 'clients' for appliance market and efficiency data – the consumers who purchase appliances and the policymakers who seek to shape the market environment[1].

Providing information for consumers

By now, consumers are familiar with the comparison shopping provided by online retailers. Retailer sites allow consumers to compare product models using a variety of features and criteria, and sort by these. To date, this type of cross-comparison has not focused on energy efficiency as a consumer value[2]. One reason for this is that efficiency is a technical parameter that is not well understood by retailers or consumers, and needs some interpretation in order to make comparisons between similar products or to estimate likely impacts on household energy use and expenditures. The general questions that efficiency comparison tools can answer for consumers are:

- Which is the most efficient available appliance in a particular class, and how do others compare in terms of performance and price?
- What are the benefits to the household of buying an efficient product? How much less energy per year will it use? What will be the electricity bill savings?
- What is the trade-off of purchasing efficient appliances, in terms of metrics such as life-cycle cost, payback period or cost of conserved energy?

Providing information for policymakers

While there is a benefit to consumers of a tool that compares individual models in a snapshot of the market at a given time, additional value is to provided to policymakers

[1] Additional stakeholders, including manufacturers and utilities might also make use of these data.

[2] Efficiency-focused sites, such as "Top Ten" do exist, but present only a small subset of models

by a data source that (1) provides the distribution of efficiency of the market at a given time (2) relates retail prices to efficiency over a large sample of models, (3) allows for comparison with other markets and (4) traces the evolution of efficiency and price over time.

Some of these datasets have been obtained in larger markets in the past at considerable expense in both money and time. A 'web-crawling' system that assembles these datasets automatically could potentially allow for much more widespread access to such datasets, including in developing countries, where access has been limited by lack of financial resources. Some of the areas that the establishment of an appliance efficiency database could facilitate are:

- Benchmarking – the complete distribution of efficiency provides policy makers with a concrete assessment of where they are at the starting point of program implementation and how their market compares to historical data or those in other countries. This information helps to set goals and design programs.

- Certification Checking – A straightforward but useful compliance function is to combine data collected from online retailing sites with certification databases in order to ensure that models offered for sale are properly included in the certification database.

- Market Monitoring - because of the relatively low-cost and time expenditure needed by an online database, it becomes feasible to periodically sample data and trace the evolution of the market. This provides detailed and rapid feedback on program design and can serve as the basis for policy impacts evaluation.

- Price evolution – the availability of a sample of both price and efficiency at different points in time allows policymakers and researchers to better understand the impact of market shifts on prices, e.g., through economies of scale and pricing policy.

- Standardization – while there are many difficulties in comparing the diverse range of product energy efficiency data across markets, there is also a challenge in aggregating and analyzing data in a single market from the various government agencies, certification bodies, and retailers.

A database framework setting forth generic data structures for appliances is a fundamental first step toward creating the valuable tools described above. Their ultimate success will rely on the applicability of these structures across end uses and markets. The objectives described in the next section are set with this in mind.

Project Objectives

The main objective of the Data Access project was to develop an international energy efficiency data standard enabling wider exchange of energy use, ratings/certifications, and pricing data, and apply it to two specific product categories (Room ACs and TVs were selected). The framework would be applicable to a wide range of other product categories as well.

The key goals for developing a global framework for capturing energy efficiency and pricing data on appliance products are the following:

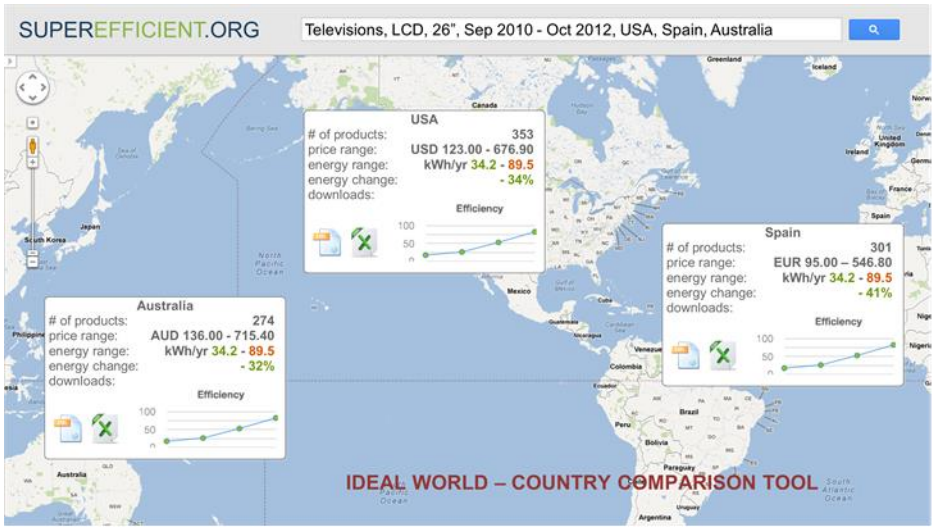
- Single product lookup for energy use, ratings/certifications, and pricing data
- Standard data model that enables developing countries to reference certification systems of their larger trade partners as a basis for their own programs
- Web and mobile apps to facilitate purchasing of more efficient products

Enervee and LBNL were asked to envision potential applications of an implementing global framework to provide SEAD members with insight into the end use applications. After consultation with CLASP, two "ideal world" scenarios that were developed to showcase the potential of the framework for both consumer and policy related use cases.

Figure 1: Interactive Mobile Platform – mockup of potential application

1) Interactive Mobile Platform to look up price, energy efficiency, certifications/ratings, and potentially coupons and other special offers.

Figure 2: Cross-country Comparison Tool – mockup of potential application



2) Cross-country reporting framework to compare pricing and efficiency information across products in different markets.

Project Methodology

Enervee and LBNL used a structured approach to gather requirements and develop the data framework and data standard, identifying the following as key tasks to complete:

- Gather requirements and analyze data sources
- Develop prototype data framework and data standard
- Conduct use case testing with SEAD members using live data
- Revise data framework and standard based on feedback

Process for Creating the Global Data Framework and Standard

The entire process followed an iterative model with stakeholders being consulted at each step of the process. This ensured that each participating country's specific requirements were incorporated into the model.

Gather Requirements and Analyze Data Sources

One of the main challenges in developing a first of its kind data framework is that potential users do not have a clear idea of how it will be used. To gain a better understanding of how SEAD members are currently using their respective certification and rating databases, specific data was gathered about current/future uses and challenges faced using an online survey.

Enervee and LBNL worked with CLASP to reach out to as many potential SEAD members as possible via the SEAD working group calls. Below is the list of stakeholders that provided data directly for use in the project.

Table 1: Stakeholders

Organization	Country
Swedish Energy Agency	Sweden
Korea Testing Laboratory	South Korea
US Department of Energy	USA
Natural Resources Canada	Canada
Department for Environment, Food & Rural Affairs	UK
Department of Climate Change and Energy Efficiency	Australia
IEA 4E Mapping and Benchmarking Annex	UK

Based on the feedback received, we noticed that there were two distinct types of use cases. The first was for regulatory purposes ("Policy") with the end users being government organizations or associated nonprofits/agencies. The second was focused on the purchasing experience ("Consumer") with the primary end users being consumers and businesses. The use cases are summarized in Table 2. Those that were identified in the stakeholder process as *high priority* use cases were fully included as part of the framework. *Low priority* use cases required additional work that is out of scope for this project.

Table 2: Use Cases

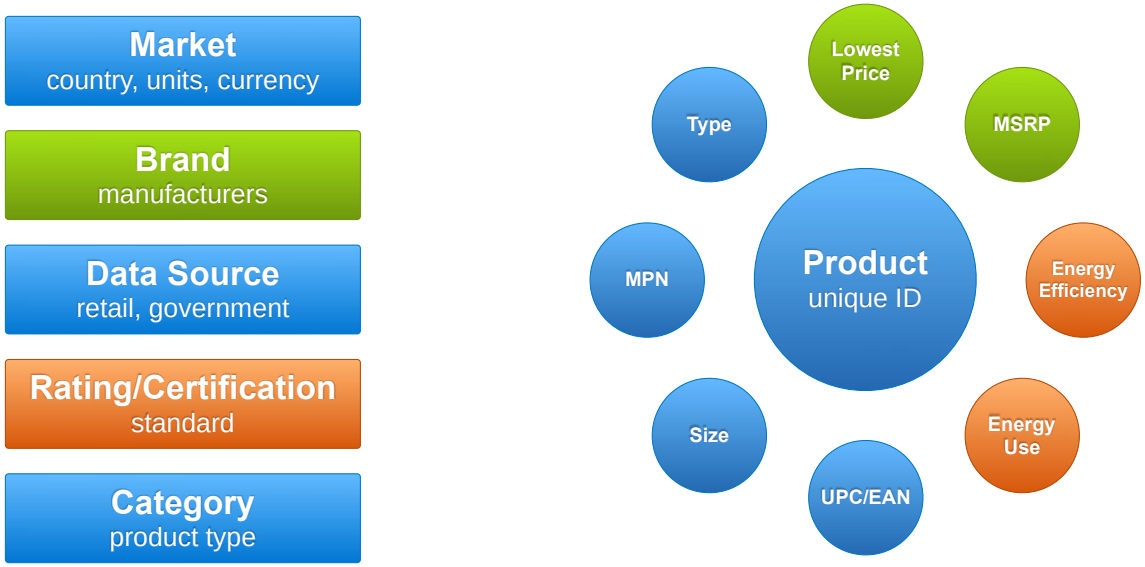
#	Use Case	Need Type	Priority
1	Access to other countries' certifications and ratings for monitoring, verification, and policy design	Policy	High
2	Connect sales and efficiency trends	Policy	High
3	Ability to assess impact of marketing campaigns on consumer adoption	Policy	Low
4	Enables comparison of product rating methods across countries	Policy	High
5	Facilitates procurement activities	Consumer	Low
6	Enables development of consumer mobile apps/online comparison tools	Consumer	High
7	Facilitates product energy efficiency awards (Energy Star & SEAD medals)	Consumer	Low
8	Facilitates utility incentive programs	Consumer	Low
9	Enables comparison of testing procedures	Policy	Low

Develop Prototype Framework and Data Standard

After analyzing the data sources provided in stakeholder surveys, the Enervee and LBNL team developed a prototype database framework that would be able to leverage existing data sources but also take advantage of richer product data from online shopping APIs (Application Programming Interface).

The prototype data model was product-centric and allowed for the capturing of both product specifications and energy consumption/efficiency attributes. It was designed to standardize brands and units of measurement while also tagging each data object with its respective source.

Figure 3: Prototype Framework



Conduct Use Case Testing with Live Data

To test the framework's capacity to support the high-priority use cases, the Enervee-LBNL team chose the US, Australia and South Korea as test markets to populate the prototype database with data on TVs and Room ACs for sale in each market.

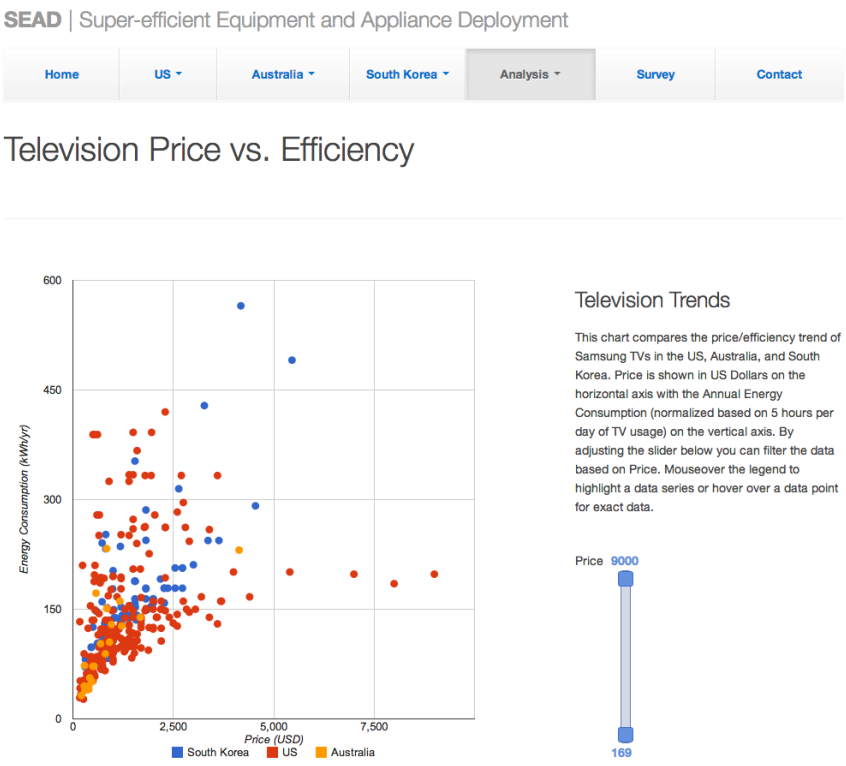
Enervee captured data from the following sources:

Table 3: Data Sources

	Australia	South Korea	US
Product Pricing & Specs	• Ebay	• EELSP certification data	• Amazon • Best Buy • Ebay
Energy Efficiency	• E3 Energy Rating	• EELSP certification data	• Department of Energy • California Energy Commission • Federal Trade Commission

For Australia and the US, product specs and pricing were pulled directly from online shopping APIs. Energy efficiency information was then matched to these products based on model number. For South Korea we did not have access to an online shopping API and therefore used pricing specs and energy efficiency data from the EELSP government certification data set.

Figure 4: Use Case Testing Price-Efficiency Analysis



Besides setting up a consumer portal to compare products based on price, product specs, and energy efficiency, there was also a demonstration of cross-market analysis for policy making. Four data views were developed:

- Room AC Price vs Sales
- Room AC Price vs Efficiency
- TV Price vs Energy Consumption (As shown in Figure 4 above)
- TV Screen Size & Energy Consumption 2012 vs 2013

Enervee and CLASP arranged one on one calls with each participating SEAD member to gather feedback. A survey was also developed to capture additional feedback to feed into the revised framework.

One on one demonstrations were conducted with the following stakeholders:

- NRCAN, Canada
- Department of Resources, Energy and Tourism, Australia
- Korea Testing Laboratory, South Korea
- Department of Energy, US
- Department for Environment, Food and Rural Affairs, UK

Revise Framework Based on Feedback

Enervee spent significant time analyzing feedback from stakeholder interviews, surveys, and working group discussions to finalize the framework and global data standard. Feedback was gathered during the prototype review and use case testing process that led to a number of meaningful revisions to the model. The proposed changes and revisions to the model are discussed in detail in the "Final Database Schema" section of the paper.

Overall the stakeholders were satisfied with the capabilities of the framework for the identified use cases (Refer to Use Case Testing Results in Appendix) and the following general comments were made:

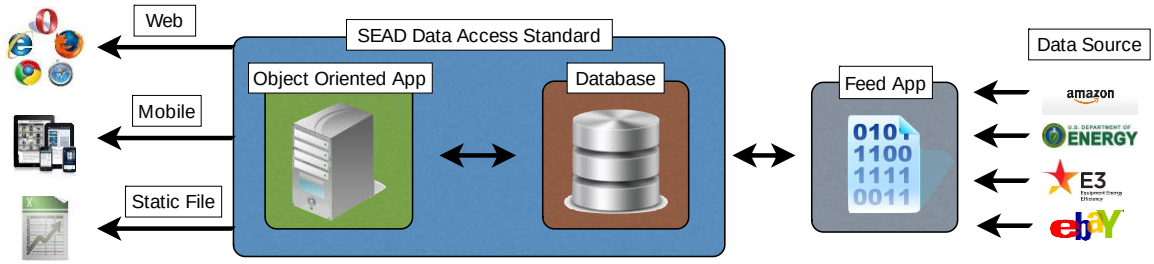
"I am happy to record that the demonstration you took me through was very helpful; what you have done to date looks really good and meets (and probably exceeds) the expectations I had."

"I think this is very promising and taking into account the way you collect data, this could be a very powerful tool both for consumers and - even more so - policy makers. The possibility to get fresh data on price, EE and sales volumes allow both for a more dynamic minimum energy performance standards setting, as well as evaluation of the same (MEPS) over time."

Final Framework & Data Standard

Overview of Data Framework

Figure 5: System design of SEAD Energy Efficiency Data Access framework & standard



The SEAD Energy Efficiency Data Access framework was designed to be able to capture and compare energy efficiency product data for the many types of consumer appliances that are sold around the world. It can be used for a single market or multiple markets.

The framework is designed to receive product data from online retailers and certification organizations, apply automated procedures to normalize and vet the data, and output the data to a web/mobile interface or static file such as an Excel spreadsheet.

It uses an object oriented, product-centric model with an extensible design to ensure that it can meet the requirements of any market or category. To address the complexity and ever-changing nature of retail product data, the SEAD Data Model is based on highly integrated, abstract data objects.

While a relational data model provides an understanding of the underlying data by looking at the relationship between tables, an object oriented model has an abstract software layer sitting on top of the table definitions that creates aggregate units of data or objects via combinations of tables and related code. These objects allow for a much more flexible and extensible database system.

Since the relationship between products, offers, feeds, data set uploads, and crawled uploads is complex and utilizes a large number of tables, it is not recommended to attempt to implement the data model in a piecemeal fashion. Each product category having a different set of attributes to capture requires flexibility in defining attribute types by category.

The core of the framework standardizes regions, brands, category taxonomy, product attributes, and units of measurement with common definitions. An intelligent application layer would connect with the framework to normalize, aggregate, and match data across the disparate energy efficiency certification and online shopping data sources.

Figure 6: Schematic diagram of the SEAD Energy Efficiency Data Access framework

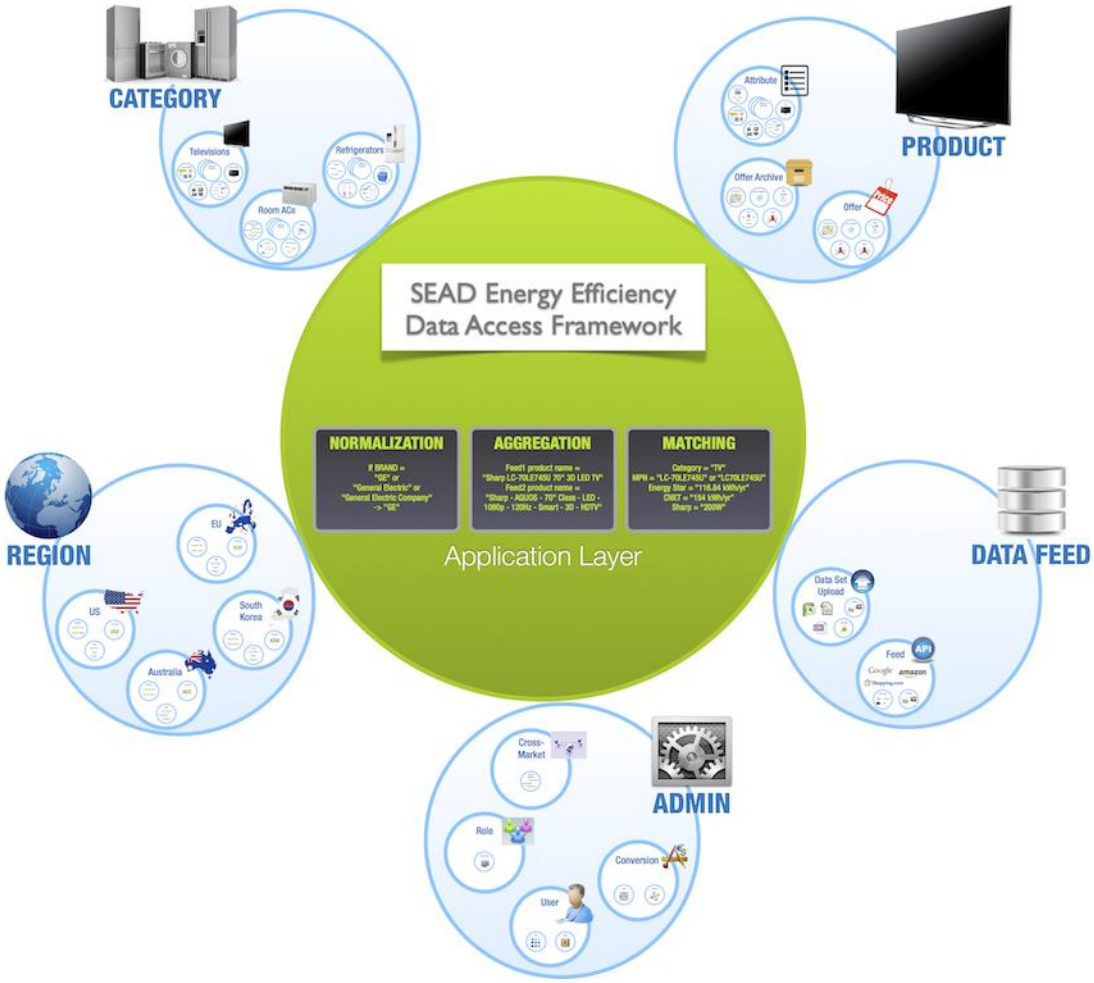


Figure 6 shows a schematic diagram of the framework. The five conceptual modules that make up the framework are:

- Region – defined as a geographic entity (typically a country) that utilizes one currency, similar units of measurement (i.e. Metric or English), product category definition, and product mix.

- Category – a grouping of products with a defined set of product attributes. A product Category can be a broad or narrow definition depending on the granularity of the product criteria. (i.e. Air Conditioners vs Room Air Conditioners)

- Product – a unique record defined by a manufacturer part number/model number (MPN) and GTIN such as a UPC-A (12 digit) or EAN/UCC-13 (13 digit) barcode. Product objects in the framework contain attributes that store information on marketing images/descriptions, features, technical specifications, prices, and energy efficiency certifications.

- Data feed – a source of data containing product or certification details that is received into the system via online shopping APIs or flat file uploads of government certification data.

- Admin – universal configuration settings for geo-location, unit conversions, currency exchange rates, brands/manufacturers, and access rights.

The intelligent application layer in the center of Figure 5 facilitates the following processes:

Normalization

All data ingestion is controlled by application level software responsible for the filtration and conversion of units of measurement and the normalization of brand names.

To assist in doing this, the architecture is dependent on having the ability to create complex composite data types at the application layer that translate into separate database columns. Access to the entire grouping as well as the individual member columns or fields will be necessary.

A concrete example is the composite type used to store a numeric value and the corresponding units of measurement. The application considers the pair to exist as a floating point value and string units but stored as separate database columns. Moreover, the implementation should permit separate addressing of the columns through the composite type.

Aggregation

Actual products are obtained by polling online shopping APIs on a daily basis and then aggregating the results. The key to this process is the ability to uniquely define product data in such a way that it is possible for data on the same product model coming from

multiple APIs to be mapped to a single product by way of UPC, EAN, or MPN codes This process will invariably be a sequence of database calls and results analysis.

Application level software is dependent on the normalization processes during product discovery but also responsible for product creation as well as updating product attributes. As the architecture breaks up the representation of a product into various connected objects or related tables, the process depends on application level software to maintain these objects consistently.

Matching

To further improve data quality, additional data sets are mechanically matched against existing product data, which is then updated in a variety of ways. For example, a product's energy use, which frequently is not available from online shopping APIs can be obtained from government certification databases that have been matched to the product data. All other data is matched as attributes which are sets of key/values pairs further describing the related product.

The match process requires a combination of application logic and database functions in order to do MPN matching at scale. Since MPNs provided from certification data sources are not always exact and could include wildcards, the architecture must include methods of improving the match rate. With wildcards, it is important to ensure that matches are actually valid and not due to the loosening of syntax.

Software Domain Design

A detailed description of how each of the applications within the data framework and the individual objects that compose it is documented in the Appendix in the Software Domain Design section.

Global Data Standard

An essential part of establishing a data standard is agreeing on a simple, consistent model that can be applied across diverse markets. To accomplish this task, the concept of a Global Category was developed to define a universal set of attributes that are gathered irrespective of the product category. This is the default information that will be captured, if available, for any category, covering the unique product identifiers, market, brand/manufacturer, pricing, energy, certification, and marketing related data.

Global Category Definition

To the greatest extent possible, the Global Category definitions have been based off of the IEA 4E Mapping and Benchmarking and revised based on SEAD member feedback. These definitions were chosen to facilitate harmonization of product category definitions across regions using international standards. Figure 7 and Table 4 show the proposed definition.

Figure 7: Global Category Data Standard

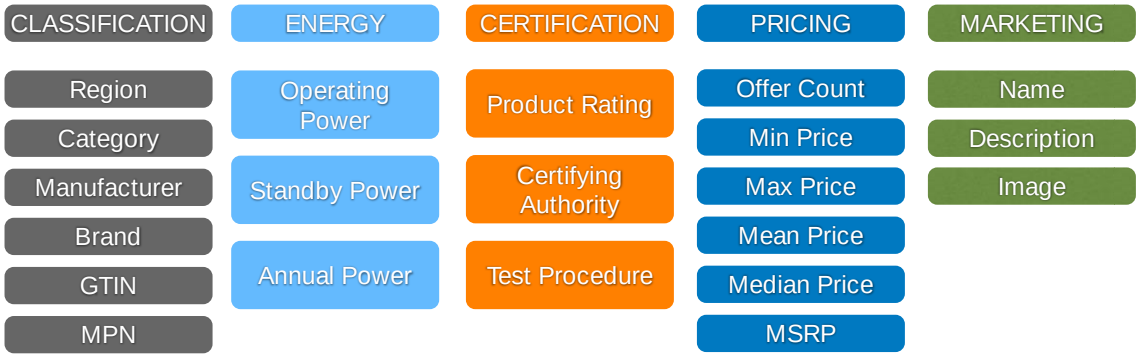


Table 4: Global Category Data Standard

Attribute	Data Type	Notes
Manufacturer Part Number(s) (MPN)	String	Product Model Number stripped of extraneous characters
Brand	String	Normalized Across All Markets
Manufacturer	String	Parent Relationship to Brand
Region	String	Defined region based on ISO 3166-1 country codes. Reference to base_region
Global Trade Item Numbers (GTIN)	String	Universal Product Code (UPC), International Article Number (EAN) or QR Code
Price	Numeric	Lowest Current Price Max of all Offers Avg of all Offers Median of all Offers Standard Deviation of all Offers
MSRP	Numeric	Manufacturers Suggested Retail Price

Energy consumption metrics	Numeric	Operating/Annual Consumption in W or kWh Standby, and Sleep in W. Can capture multiple values from different sources.
Product Certification information	String	Test Procedure, and Certifying Authority associated with energy data
Product Images	String	Links to Images of the Product (multiple)
Product Marketing	String	Product Title, Headline, and General Description

Product Category Definition: TV and Room AC

While the Global Category attributes capture the common data across all products, it is also necessary to capture attributes specific to each product category, since the data required for TVs will vary substantially from that of Room ACs or Refrigerators. Tables 5 and 6 list the product attributes that make up the proposed universal TV and Room AC category definitions. Additional attributes can be added on a region-by region basis, but these category-specific attributes are the descriptors that bridge regional differences across markets and enable cross-market comparison.

Figure 8: TV Data Standard

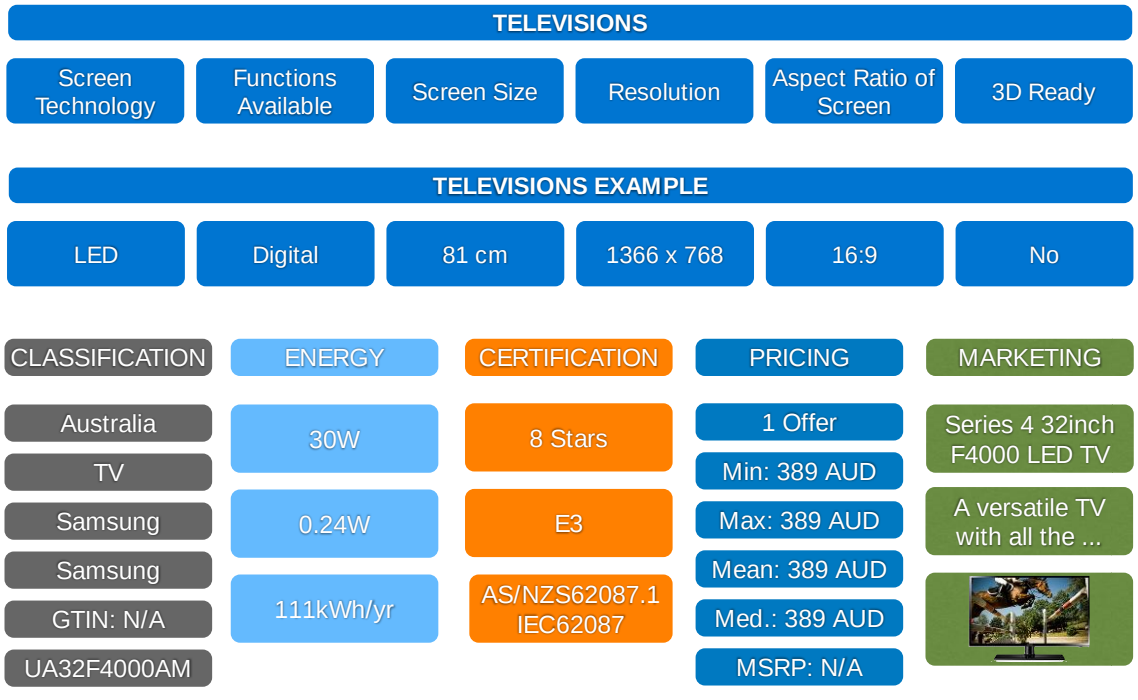


Table 5: TV Data Standard

Attribute	Type	Notes
Screen Technology	String	LCD, LED, Plasma, Projection, CRT
Functions Available	String	Analog, Digital
Screen Size	Numeric	Size in inches, centimeters, meters, or other unit of measurement
Resolution	String	HD or Conventional
Aspect Ratio of Screen	String	4:3, 16:10, 16:9
3D Ready	String	Yes, No

Figure 9: Room AC Data Standard

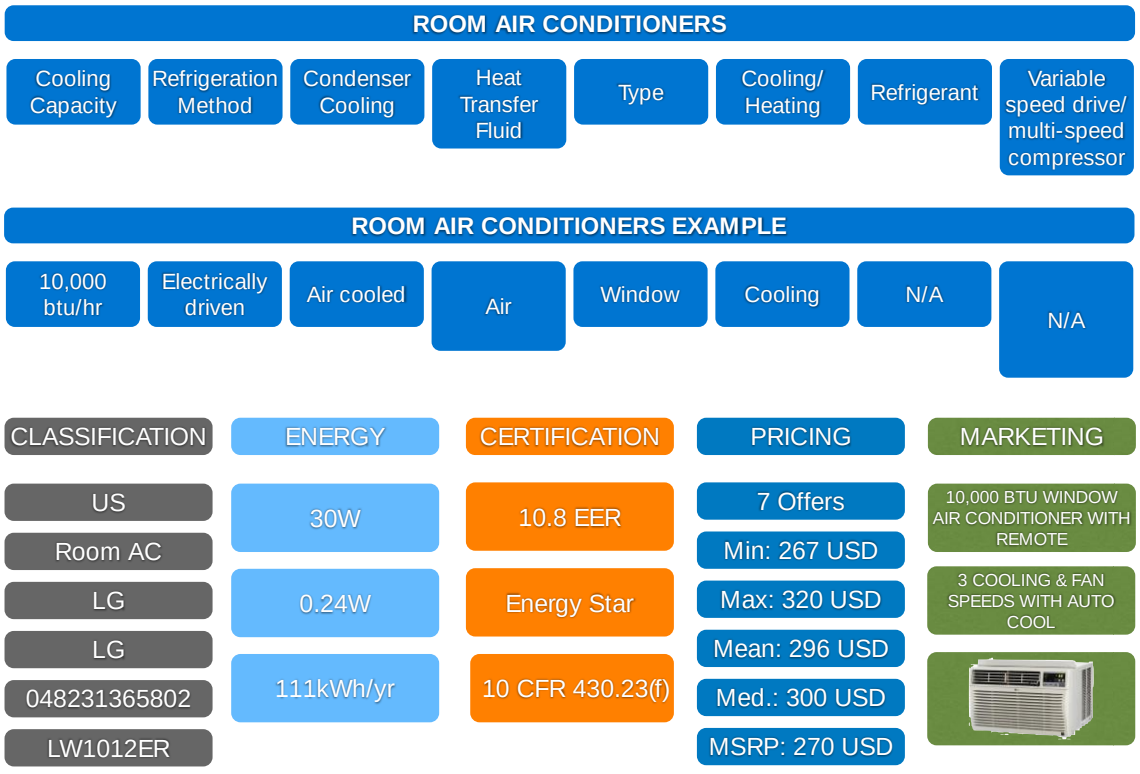


Table 6: Room AC Data Standard

Attribute	Type	Notes
Cooling Capacity	Numeric	BTU, kW, W
Refrigeration	String	Electrically driven,

Method		Absorption units
Condenser Cooling	String	Air cooled, Water cooled
Heat Transfer Fluid	String	Air, Water
Type	String	Unitary, Split units, Multi-split
Cooling/heating	String	Cooling, Heating, Cooling/heating
Variable speed drive/multi-speed compressor	String	Present or Not
Refrigerant	String	Refrigerant Type

Initial Database Schema

The initial data schema was built based on the requirements gathered in Phase I from SEAD members. It was a product-centric model and was designed to handle an unlimited number of products, markets, and data feeds.

The data schema for the SEAD Data Access framework was based off the guidelines below:

- Product records can only be created with a valid MPN or GTIN code. If when adding new product records a duplicate MPN or GTIN code is found in the same category and region, the product attributes and offers will be aggregated with the existing product record. Product attributes will only be updated if the data priority for the new data source is greater than or equal to the data source on the existing attribute. This prevents a database based on the framework from being populated with duplicate product records. The framework assigns a unique ID to each product.

- A product typically has a last known price but may not always have a current price (e.g., if it is discontinued or out of stock). It could have one or many prices offered by different merchants.

- A product is assigned to a single category, single brand/manufacturer, and a single region (based on ISO 3166-1 code). Product categorization is flexible, in that a category can be defined as broad or as narrow as is required. For example, a category could be defined for Air Conditioner including both central and room air conditioners or there could be two categories defined for Room Air Conditioner and Central Air Conditioner. While the framework supports

flexibility, categorizations should be agreed upon on at an international level to facilitate a global data standard.

- Brands are normalized globally and allow for a parent-child hierarchy to capture both the consumer brand and the manufacturer or parent company.

- All data captured in the system is date stamped and tied to a specific data source. Data sources are online shopping APIs or government certification and other data uploads such as web crawling. This makes it very easy to identify when data was inputted and where it came from.

- Data sources are given a quality ranking from 0 to 100. This handles the case whereby a product such as a TV is available in both an online shopping source (e.g., Amazon) and government certification feed (e.g., Energy Star). In that case there is a preference to capture the product attributes (images, description, flat panel type, etc) from Amazon but the energy consumption from Energy Star. The framework is designed to capture only a single value into each attribute based on the highest priority data source. Storing multiple values for an attribute is not recommended.

- A system using the standard can apply logical rules to a data feed to standardize naming and categorization or set criteria for attribute values. Mapping of all attribute names and values to normalized values is required to define rules for each category (see Data Rules section below) within a data feed and these rules would then be applied whenever data is ingested from an API. This ensures that product data is captured and categorized in the same structure across multiple data feeds. This approach provides a straightforward way of normalizing data without requiring additional computing infrastructure or any manual editing of the data.

- Key product attributes such as UPC/EAN, MPN, Brand, Region, Energy Consumption, and Price have been developed as distinct objects, whereas general product attributes can be added as needed and are flexible and extensible.

- A product can have many attributes, with all numeric attributes having a defined unit of measurement. This ensures that when data is compared across markets, it is easy to apply conversions for currencies, sizes, and capacities, as the units are always known.

- Units of measurement are standardized. This refers to the fact that units will be stored in a normalized way such as kWh/yr for annual energy consumption, W for watts, lbs for pounds, kg for kilogram, in for inches, cm for centimeter, etc.

- A product receives a single energy consumption record, capturing the annual energy consumption and operating and standby watts. Energy consumption records are pulled from government certification data upload and matched to a product based on MPN or GTIN code.

- The framework supports capturing of new product offers as frequently as daily. Historical offers would be stored in an offer archive summarizing the count, minimum, maximum, mean, median, and standard deviation of each day's offer prices.

Final Database Schema

While the initial schema covered the main use cases of the SEAD stakeholders, it became apparent additional requirements would greatly benefit a global data standard. The following areas were identified:

- Method to capture certification and test procedures data.

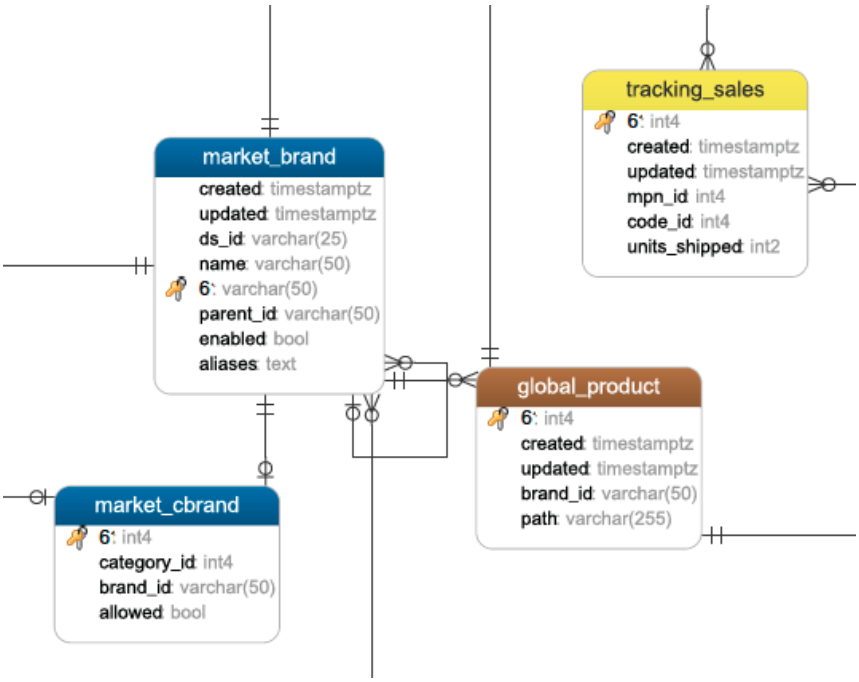
- Method for handling cross-market mapping.

- Need to capture multiple Product Energy values from more than one data source.

- Capacity to track data on product sales volume.

To address the additional requirements, revisions to the initial schema were made (sample below, full diagram in Appendix). The main change was in extending the model to support a robust global data standard. Consensus on a set of mandatory attributes for each category definition among the SEAD stakeholders is required to successfully support the data standard with the framework.

Figure 10: Diagram of a section of the database schema



The following additions were made to the initial schema:

- Global Product object to match like products that have a different MPN or GTIN across markets.
- Crawled object to enable creation of new products via a data set upload of certification or other static data sets.
- Tracking Sales object to capture monthly or annual sales at a per MPN level. This data would be imported from third party sources.
- Certification object created to enable capturing of Certifying Authority, Certification Program, Test Procedure, and Registration Date.
- Currency Exchange object to enable conversion of prices across markets. Exchange rates could be updated on a regular schedule (daily, weekly, or monthly) as agreed by the stakeholders using the data standard.

Implementation Guidelines

As the Data Access framework uses a highly abstract data model it is recommended that prior to implementation an experienced data architect completes a thorough scoping of the project requirements. To achieve a fully functional database system, an intelligent application layer must be developed on top of the underlying database to handle data ingestion from APIs and flat files, normalization of product attributes, and aggregation of offers across data feeds.

To maintain the model's flexibility and compatibility with the data standard and other datasets, it will be important to retain the following core aspects:

- A **category** module that specifies the attributes that are common to a particular product category and adheres to the **global data standard**.
- A **data feed** module through which raw product data from various sources can be imported into the data framework in a wide variety of formats and reshaped according to source-specific rules.
- A means to reliably **normalize products**, that is, identify data entries from different feeds that correspond to the same product. Possible approaches include matching on model numbers, GTIN codes, or certain key attributes.
- A means to reliably **normalize attributes**, that is, identify the names of features in different feeds that refer to the same attribute and rename them appropriately. For example, "Size" and "Diagonal Size" might be the name given to a television's "Screen Size" global attribute in two different feeds. Resolving this is most naturally done by means of rules specified in the feed module.
- A **product** module to store the data from the feed module after identical products and product attributes from different feeds have been cross-matched and normalized. This module should store a core set of data about each product, including the normalizing information, as well as a flexible and extensible means for storing arbitrary additional attributes as defined for the associated category.
- A **market** module, if the use case requires product data to be collected and normalized across different markets.

- **Administrative tools** for defining categories, feeds, and normalization rules.

Deployment Options

There are two options for deploying the framework globally:

1) A single instance that is operated and maintained centrally for each of the participating countries. This would require coordination of requirements and timelines across different organizations but would significantly cut down on implementation and operation & maintenance costs. It would also be the easiest way to facilitate the global data standard.

2) Multiple instances for each participating country. This would allow for customized implementations at a per country level, but would have higher implementation and operation & maintenance costs. It would require a higher level of supervision across countries to ensure that compliance with the global data standard is maintained.

Hosting and Infrastructure Considerations

It is recommended that the data framework and application layer be deployed on a cloud based hosting environment such as Amazon Web Services. This provides the ability to scale resources on demand with global redundancy and a high level of data security.

To provide guidance on the level of investment required to host the system on a cloud based server, it costs approximately $1,000/month for a market the size of the US with 10+ product categories of data. To host the framework at a global level covering North America, EU, and Asia a maximum budget of $10,000/month would be required.

Database and Framework Setup

As part of the final report, four files have been included to enable easy installation of the database with 300 products of sample data for Australia, South Korea, and the US markets.

- SEAD Data Access Final Data Schema.sql – the full set of tables and indexes required for the data framework
- SEAD Data Access Sample Data.sql – sample product data for Room ACs and TVs for the Australia, South Korea, and the US markets.
- SEAD Data Access Schema Documentation.txt – definition of all database tables with field types, primary and foreign keys

- SEAD Data Access Final ER Diagram.pdf – visual diagram of table relationships for the entire framework

The first step is to run SEAD Data Access Final Data Schema.sql in your database environment – Enervee recommends PostgreSQL. Once the schema has been created, it is recommended to run SEAD Data Access Sample Data.sql to add sample data. This will provide enormous help in understanding the object structure and relationships.

To enable the normalization, aggregation, and matching processes described in the previous section, it will be necessary to utilize an ORM (Object Relational Mapper) such as Python/Django. Objects should be constructed for each database app as detailed in the Appendix.

Once the objects have been constructed, the final step is to implement the application logic to process data feeds to create products and match energy consumption and efficiency certification profiles.

Customization by Market

While the standard sets guidelines for which attributes to capture, it still allows for flexibility when implementing for a specific market. Each market using the framework will have the capability to add new attributes to an existing category or define a new category for products that are not currently covered. These extensions cannot be compared across markets unless they are fully adopted into the global data standard.

Extending a Category Definition

For example, some countries may want to classify types of Room ACs in more granularity (window vs. through-wall units, for example) and therefore there may need a 'SubType' attribute added to their Room AC category definition. Additionally, there may be a need to set up an attribute to capture a numeric rating that is assigned by a country's certification program.

New Category Definitions

It is also possible to define an entirely new category that is not currently covered by the data standard. This new category would automatically inherit the base attributes (such as MPN, GTIN code, and Brand) that are part of the Global Category definition and would only need to have specific attributes added that are unique to that product category and market.

As more and more countries begin to adopt the data standard, the result will be that the majority of categories will have been standardized across all markets. To ensure that the data standard remains a common framework internationally it is recommended that a single organization facilitates the update process and publishes new versions of the standard online.

Application Layer: Populating the Data Framework

To make the adoption of the data framework as seamless as possible, a comprehensive approach for populating the framework from existing data sources is detailed below. An application layer is required to normalize the disparate data across markets into the new framework.

A system using the framework would create unique products from online shopping APIs such as Best Buy, Amazon, and Ebay by pulling in product IDs, marketing details, technical specifications, and prices. It would then match energy efficiency data based on MPN or GTIN via uploads of certification data sets. An unlimited number of feeds from online shopping APIs and certification data sets can be utilized in the framework to ensure strong coverage of each market.

Strategy for Utilizing UPC or QR Codes

The framework relies on having either an MPN or GTIN code to create a unique product. These codes provide a way of vetting that a product is unique based on its Brand, Category, and Market.

There is the capability to capture multiple MPNs for a product as there are often variations based on retailers adding "/", "-", spaces, or other extraneous characters. Multiple GTINs can also be associated with a product such as a UPC (Universal Product Code), ASIN (Amazon Standard Identification Number), or EAN (International Article Number). These codes are captured directly from the retail APIs into the Product Codes object. Both Ebay and Amazon have very good coverage of this data.

QR (Quick-Response) Codes can also be captured into the Product Code object from a retail API or from a data set upload. The framework could be connected to a barcode-scanning tool to enable the look up of data on physical products via a mobile application.

Importing Products into the Framework

There are two ways to get data into the model: 1) using an online shopping API with the Feeds object or 2) using a static data file with the Crawled object. Using an API is a more robust and automated way to create and update products, but in cases where this is not possible a static data file is fully supported.

The full methodology and technical details for importing products into the framework can be found in the Appendix.

Matching Energy and Attributes via Certification Data Sets

Government certification/rating data is utilized to capture products' energy efficiency information. This data is imported into the framework from a static data file. An application layer then matches energy efficiency data to existing product records. This is accomplished using the methodology below:

- Data set parsing: certification data sets are uploaded to add energy efficiency related attributes to product records
- Matching: associated energy efficiency data, such as energy consumption, certification level, certifying authority, registration date, etc., is captured for already existing products based on a matched MPN or GTIN code.

If an MPN match is not successful there is also the capability to capture products from the certification record directly into the database via the Crawled App. This will cover the use case in which certification data sets contain historical products that are not currently available on the market.

Recommendations on Certification Data Sets

The Enervee team found a number of challenges in handling the wide variety of formats and naming conventions used in current certification data sets. The following list of practices would more easily facilitate moving to a common data standard:

- Capturing individual product MPNs (model numbers) for certification records instead of using wild cards to enable a single certification record to apply to multiple products. As MPNs can have differing formats based on the retailer (such as additional hyphens or slashes) it simplifies matching of energy records to product pricing data.

- Capturing UPC/EAN for each product in certification records would enable a more straightforward way of linking certification data with retail data.

- Normalization of Manufacturers/Brands at an international level would simplify cross-market matching and trend analysis across countries. This would be accomplished by creating a list of aliases for each Manufacturer/Brand across all the variations present in online shopping APIs and government certifications data sources. It would be one time effort to create the normalized list and then a small amount of maintenance would be required when new APIs or certification sets are used.

- For any certification data sets that capture a product's annual energy it would be extremely helpful to explicitly declare the usage assumptions and test procedures used within each of the certification data files.

- Normalizing the way units of measurement are written at an international level (i.e. defining the acceptable list of units and standard abbreviations for each unit). The framework would allow any numerical value to be accepted as long as it includes an acceptable unit of measurement. This would eliminate significant time and effort currently required to do data parsing and cleansing. It would also facilitate easy the addition of automated conversion factors between units.

Next Steps

This current project was successful in creating a database framework that could be adaptable to use across countries and could potentially lead to an internationally accepted data standard. In addition, the project involved significant consultation with stakeholders in various countries, which informed the final project deliverables as well as shaping future objectives. The additional value of this achievement will hopefully be explored further through expansion and refinement.

This project has resulted in a prototype that already provides consumers and policymakers with significant utility for two appliance types – room air conditioners and televisions. The data for each of these products provides a model-by-model comparison of efficiency and price, and answers the first important consumer question by identifying the most efficient models and providing a calculation of the energy consumption of each option. This information allows consumers to make informed choices about their appliance purchases, thus providing a value not only to consumers, but also accelerating market transformation towards more ecologically sound technologies.

On the consumer side, much of the additional utility to be added involves expanding the dataset in terms of countries, product types and retail outlets. In addition, alternative ways to communicate cost-effectiveness parameters could be explored.

The full potential of the data access project to policymakers may be even more significant to consumers. Follow-on projects to develop this potential include:

Improved correlation between models and sales

In order to accurately monitor markets using retail data, which is by model, a weighting by model sales is necessary. Algorithms and supplementary datasets that could allow for this weighting could improve accuracy and provide parity with existing datasets.

Key next task: Research and develop the capability to forecast market penetration given currently available data via online shopping APIs and government data sets.

International model cross comparison and test procedure conversion algorithms

A major advantage of a data standard is the ability to perform a cross-comparison of price and efficiency by country. This includes the ability to use empirical data to develop test procedure conversion algorithms that would then allow for benchmarking across countries. By developing attribute thresholds based on known variations in test procedures, there is the potential to automatically map like models across markets.

Key next task: Research and develop algorithms to map comparable models across more two or more markets. Test procedures for the same model between each of the markets can then be compared to determine variations.

Certification pilot projects

An international database of products sold in major markets potentially affords the possibility of small countries using the certification systems of their larger trade partners as a basis for their own programs, thus avoiding the infrastructure and resource requirements of building their own certification systems.

Key next task: Identify a market that currently does not have a certification program and sells products from markets that are already captured in other registered product databases. Leverage ratings from existing markets to create a registered product database for the new market.

Glossary

API – Application Programming Interface specifies how some software components should interact with each other.

ASIN – Amazon Standard Identification Number is a 10-character alphanumeric unique identifier assigned by Amazon.com and its partners for product identification within the Amazon.com organization.[3]

CSV - comma-separated values (also sometimes called character-separated values, because the separator character does not have to be a comma) file stores tabular data (numbers and text) in plain-text form.[3]

Database schema - a way to logically group objects such as tables, views, stored procedures etc. [3]

EAN – International Article Number (originally European Article Number). A 13 digit (12 data and 1 check) barcoding standard which is a superset of the original 12-digit Universal Product Code (UPC) system developed in the United States. [3]

ER Diagram – Entity-Relationship Diagram is a data model for describing a database in an abstract way. [3]

GS1 - is a neutral, not-for-profit, international organisation that develops and maintains standards for supply and demand chains across multiple sectors. [3]

GTIN – Global Trade Identification Number an identifier for trade items developed by GS1. GTINs may be 8, 12, 13 or 14 digits long, and each of these 4 numbering structures are constructed in a similar fashion, combining Company Prefix, Item Reference and a calculated Check Digit. GTIN-12s may be shown in UPC-A, ITF-14, or GS1-128 bar codes. GTIN-13s may be encoded in EAN-13, ITF-14 or GS1-128 bar codes, and GTIN-14s may be encoded in ITF-14 or GS1-128 bar codes.[3]

IEA 4E – Efficient Electrical End-Use Equipment co-operating program from the International Energy Agency. Twelve countries from the Asia-Pacific, Europe and North America have joined together under the forum of 4E to share information and transfer experience in order to support good policy development in the field of energy efficient appliances and equipment.

---

[3] Wikipedia.org, September 2013.

Inherited Class - a way to establish Is-a relationships between objects. In classical inheritance where objects are defined by classes, classes can inherit attributes and behavior from pre-existing classes called base classes, superclasses, or parent classes. [3]

ISO - International Organization for Standardization is an international standard-setting body composed of representatives from various national standards organizations. [3]

JSON - JavaScript Object Notation is a text-based open standard designed for human-readable data interchange. Derived from the JavaScript scripting language, JSON is a language for representing simple data structures and associative arrays, called objects. [3]

MEPS - Minimum Energy Performance Standards.

MPN – Manufacturer Part Number. An identifier of a particular part design used in a particular industry. [3]

MSRP – Manufacturer's Suggested Retail Price is the price which the manufacturer recommends that the retailer sell the product. [3]

Object Oriented - paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. [3]

QR Code - Quick Response Code is the trademark for a type of matrix barcode (or two-dimensional barcode) first designed for the automotive industry in Japan. [3]

SEAD - Super-efficient Equipment and Appliance Deployment is an initiative of the Clean Energy Ministerial is a voluntary multinational collaboration whose primary objective is to advance global market transformation for energy efficient products.

UPC – Universal Product Code. A barcode symbology that is widely used in the United States, Canada, the United Kingdom, Australia, New Zealand and in other countries for tracking trade items in stores. Its most common form, the UPC-A, consists of 12 numerical digits, which are uniquely assigned to each trade item. [3]

XML - Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. [3]

Appendix

Use Case Testing Results

A survey was developed to capture feedback on how well the prototype met the requirements of the 3 main use cases and how well the Data Access Framework could facilitate a global data standard. Reponses were on a scale from 1 (very dissatisfied) to 5 (very satisfied).

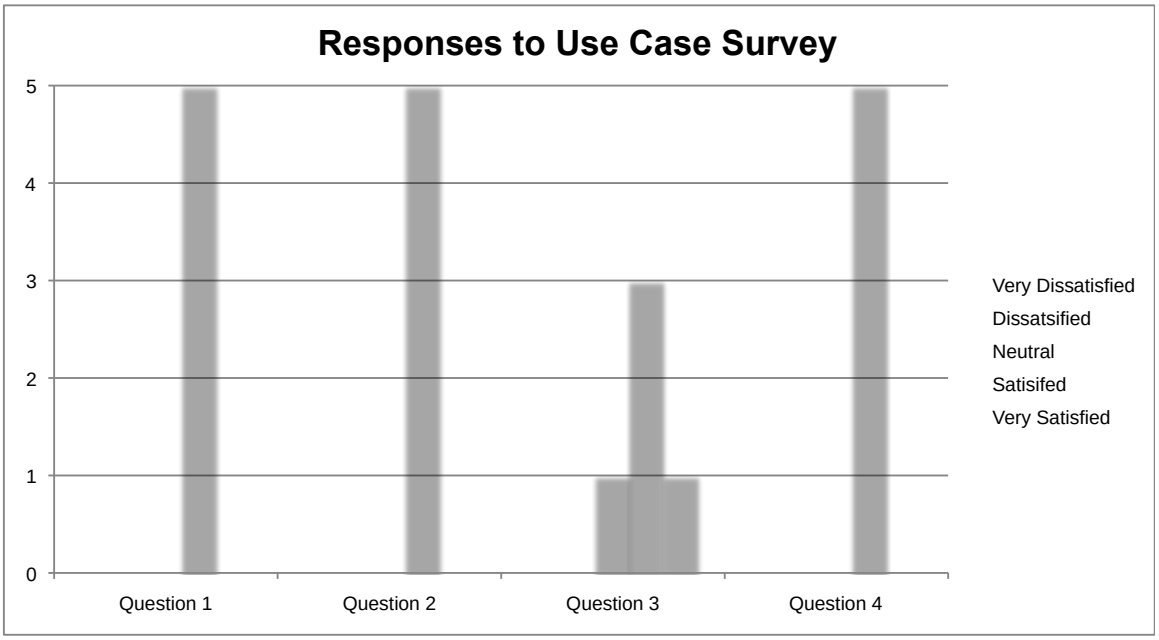
The following questions were posed:

1) How well does this prototype demonstrate that the data framework can support a web or mobile interface for consumer product comparison?

2) How well does this prototype demonstrate that the Data Access framework facilitates the ability to track and compare energy efficiency and pricing data across countries?

3) How well does this prototype demonstrate that the Data Access framework can facilitate analysis of sales data with pricing and efficiency data?

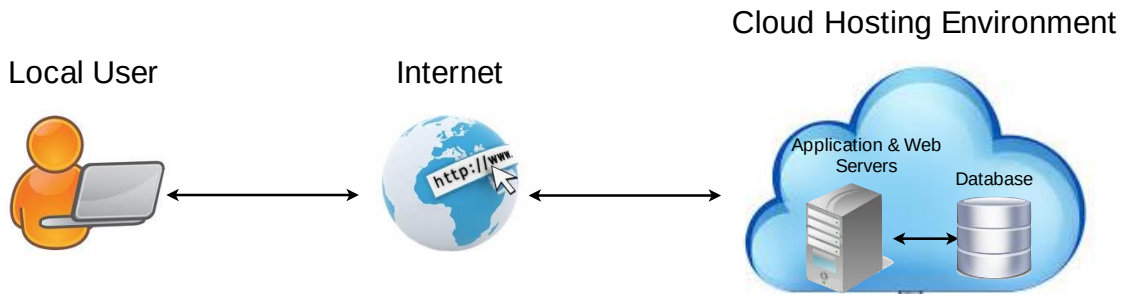
4) How well does this prototype demonstrate that the Data Access framework can facilitate a global energy efficiency data standard?



Key Takeaways:

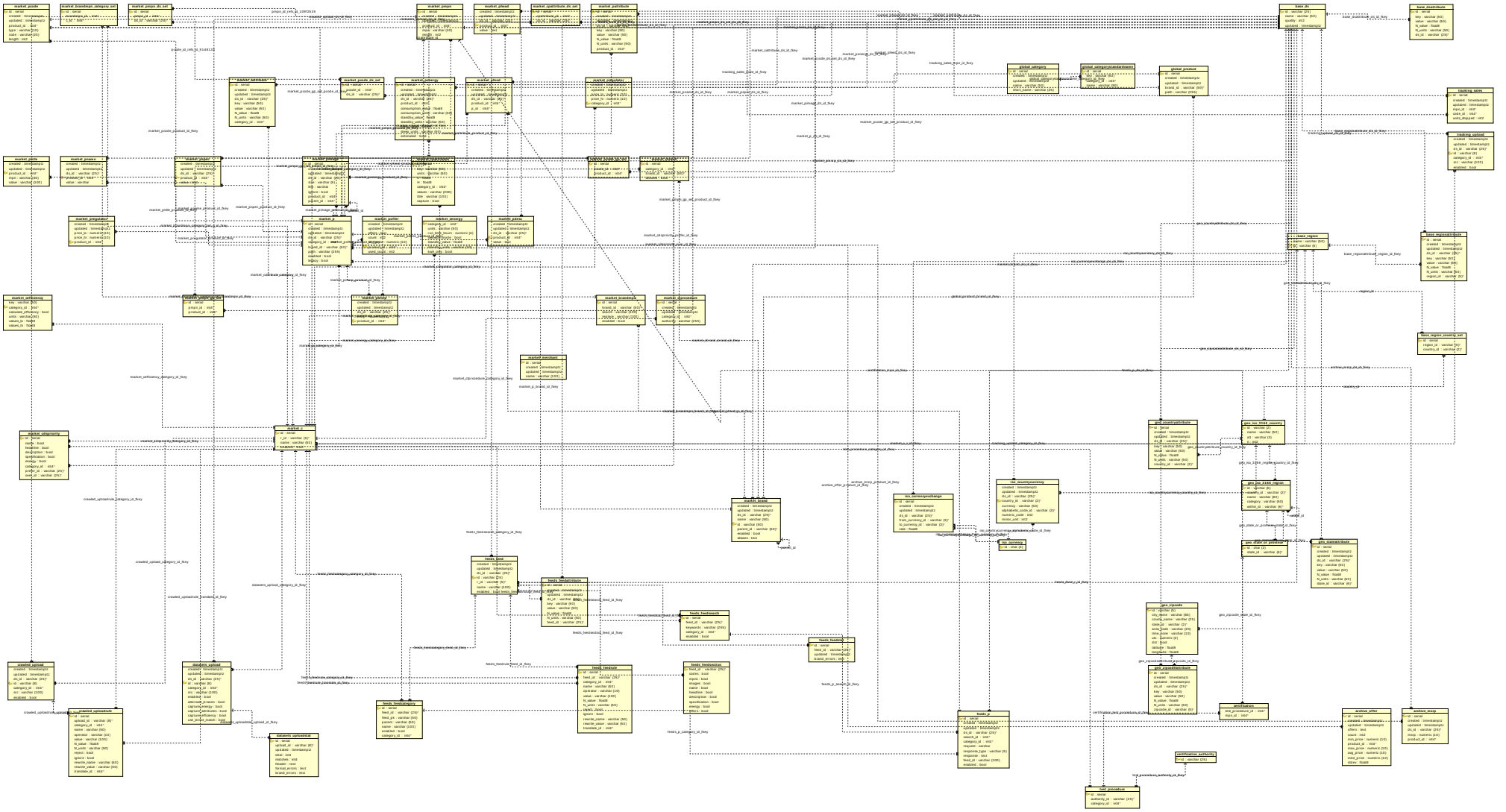
- Potential for using more than one online shopping feed to gather pricing data is very intriguing to SEAD members.
- A mobile version of the application with barcode scanning is of interest.
- Many questions were posed on what type of data was being shown in the prototype and how the user interface or data analysis charts could be customized to better present the data. This is a question of implementation not of framework design.
- Interest in being able to cover additional product categories and markets.
- Interest in including three-dimensional charts combining price, efficiency, and sales data. This is all possible once the data framework is in place and capturing daily or monthly data.
- Interest in providing consumers with ability to specify their weather zone to be able to modify energy consumption based on humidity for Room ACs

System Architecture



The SEAD Data Access framework was designed to be accessible via a web-based interface on a local user's computer. The database and application layer would be deployed in a cloud based hosting service such as Amazon Web Services. This provides for high availability, reliability, and security.

Energy Efficiency Data Access Entity Relationship Diagram



Importing Products into the Database Framework

Online Shopping APIs

Online shopping feeds from retailers with the broadest product coverage could be utilized for creating a list of all current products on the market. One or many APIs per market will be accessed on a regular basis, as frequently as daily, to pull the most up to date products and pricing data.

The methodology for importing products via online shopping APIs can be found in the Appendix.

The methodology for doing this as follows:

- Data feed retrieval: establish a connection with the online shopping API to select which products should be returned by the data feed
- Normalization: map the raw data in the retailer data feed to the definitions of the data standard. Data rules will need to be applied to normalize MPN, Brand, and product attributes.
- Aggregation: combine product specifications and offers from multiple online shopping feeds to create and maintain a single, unique product record in the framework.
- Matching: update offers and product specifications on already existing products based on MPN or GTIN. Matching MPNs requires logic to strip extraneous characters and process the many wildcards present in certification records.

Data is captured into the database from online shopping APIs first as a raw product in **feeds_P** table using the Feeds app. This contains the *Feed_id* and the *Response* which has all of the product data in the raw XML or JSON message. A separate raw product is created for each *Feed_id* that is returned from the API for each data feed (i.e. Best Buy and Amazon would both create their own raw products).

The *Created* field holds the date stamp for when the raw product was first created. Each time the feed is pulled (daily/weekly/monthly) the entries in the **feeds_P** table are updated with the latest offers. The *Updated* field then stores the date stamp for when the offers were last updated.

This means that more than one raw product can have the same MPN/GTIN code as a single feed could have duplicate products or the same product is available in more than one data feed. Raw products are created before any validation logic is applied.

The steps for processing a new shopping feed are as follows (the names of database tables are given in bold):

1. Define the category(s) in **market_c** and attributes in **market_cattribute**
2. Add feed to **feeds_feed** and feed category mappings to **feeds_feedcategory**
3. Create raw products with the XML or JSON message in the **feeds_p** table
4. Set up Feed Rules for product categories & attributes using **feeds_feedrule**
5. Categorize raw products and normalize attributes to the standards you have set up in **market_cattribute** for each category by Creating Live Products.

Setting up each Online Shopping API will require some customization to read and map the structure of the XML or JSON responses. There will also be ongoing maintenance to handle additions or changes to the API. The level of effort required for set up and maintenance depends on three factors: 1) Data Quality 2) Number of Product Categories and 3) Number of Attributes. Over the three years that Enervee has been working with Online Shopping APIs in the US market, there have been a total of 2-3 changes and none of these were drastic.

Static Data Files containing Product Data

New products can also be captured directly into the database using the Crawled app from a static file such as CSV, XML, or JSON. The product data is parsed with the **crawled_**uploadrule and then directly creates a live product.

The steps for processing a new static data file are as follows:

1. Add static data files (JSON, CSV, XML) to **crawled_upload**
2. Set up rules for categorizing products and normalizing attributes to the standards set up in **market_cattribute** using **crawled_uploadrule**
3. Create live products in **market_p** based on the rules

Feed Rules/Crawled Upload Rule

A raw product or crawled product must first pass the feed rules/upload rules in Feeds_feedrule or crawled_uploadrule before it is created as a live product in Market_p. These feed rules allow users to filter what products get created based on attributes (e.g. screen size between 10 and 90 in. or capacity < 40 cu. ft.). They also allow the rewriting of attribute names and values to normalize data across feeds.

Both **feeds_feedrule** and **crawled_uploadrule** have the same fields and use the same semantics:

1. *Operator* can be one of several logical operators "any", "<", "<=" ,"=", "in", "not" , ">=", ">"
2. *Value* is the attribute field name coming from the data feed or crawled file
3. *N_value* and *N_units* are the composite field to store the criteria for the rule. This could be a single number or could be a list of comma separated values.
4. *Reject* means to completely reject a product that meets this rule
5. *Ignore* means to skip capturing the attribute for a product that meets this rule
6. *Rewrite_name* means to rewrite the name of the attribute to something else (i.e. changing "Energy Star Rated" to "Energy Star Qualified")
7. *Rewrite_value* means to rewrite the actual value of the attribute to something else (i.e. changing "Qualified" to "Yes")
8. *Translate_id* maps the product to a different category than is specified in **feeds_feedcategory** or **crawled_upload**

Examples would be: TV Screen Technology in “LCD, LED” or Screen Size > “15 inches” or Room AC Volume Capacity <= “14 kW”. This ensures only data of the highest quality and for the desired products is captured into the framework.

Additionally, it will be necessary to translate attribute names and values from each data source to what has been defined in the data standard. For instance, “Screen Size” for TVs will always need to use the same name and values must be in numeric form with the units of measurement. This means that if the Ebay data feed for Australia refers to this field as “Size of Screen” the attribute will have to be rewritten to use the common field name of “Screen Size.”

Creating Live Products

Once a raw product passes the feed rules, it can be created as a product in the **Market_p** table with a unique product ID from the system. All of the attributes that are set up in the **Market_cpattribute** table should then be captured into the **Market_pattribute** table. The *ds_id* of the product would be based on which feed it was created from.

When processing a raw product and finding that a live product already exists by matching the MPN/GTIN, the attributes and offers should then be added to that product record. Each raw feed id is linked to the product, providing a reference to all of the raw products used to create that product.

Software Domain Design

Apps (Compartmentalization)

Tables are prefixed by their application container name:

App Name	Function
archive	capture historical pricing data
base	base (root) definitions
certification	product certifications & test procedures
crawled	product loading via crawled data
datasets	product matching
feeds	product loading via programmed API
geo	geographical tables
global	global (region independent)
iso	ISO tables (currency)
market	models products in regionalized category
tracking	sales tracking tables

Universal Concepts

Date/Time stamp
Many tables have created and updated timestamps. The semantics of these two are such that created is the timestamp of object creation while updated is the timestamp of last object save. The two fields allow data age to be computed.

Data Source (DS)
Every record in the database related to a product is provided by a specific data feed (API or static file). To ensure that data is traceable to its correct source, every data record is tagged with its respective Data Source (DS).
To define both the date and time stamp together with the DS, consider the two Django (Python) definitions:

```
class Record(models.Model):
   created = models.DateTimeField(auto_now_add=True)
   updated = models.DateTimeField(auto_now=True)
   class Meta:
      abstract = True
class Data(Record):
   ds = models.ForeignKey(DS, related_name='+', verbose_name=_('Data Source'))
   class Meta:
      abstract = True
```

Record is a simple base Django model holding the created and updated timestamp fields as well as the enforcement of update semantics on the updated field. The Data class inherits the Record class merging all the fields in the final object.
This pattern is repeated throughout the system semantically defining the necessary abstractions only once.

Composite Fields
Many of the product attributes that are captured have a specific units attached to them. Examples would be Screen Size for TVs (inches) or Volume for Fridges (cu. ft.). A composite type is used to store a numeric value and the corresponding units of measurement. The application considers the pair to exist as a floating point value and string unit but stored as separate database columns. Moreover, the implementation should permit separate addressing of the columns through the composite type.

Standardized Units
All numeric attributes in the system are associated with a unit type. This ensures that all values are always stored with their corresponding units. These unit types have been normalized, for example, inches, in., and inch are all recognized as the same unit type.

Unique Product
The framework deems a product unique based on entries in the **market_pmpn** together with **market_brand** and **market_c** (category) tables as well as the globally unique **market_pcode** table. Products are not allowed to be created in the system without at least one MPN or code and never without brand and category associations. Often times different retailers may refer to the same product using different MPN variations. In addition, refurbished products from some retailers (i.e. Best Buy) may be given a different GTIN code. This means that although technically there will not be any products with the same MPN/GTIN, there may still be duplicate products in the data.

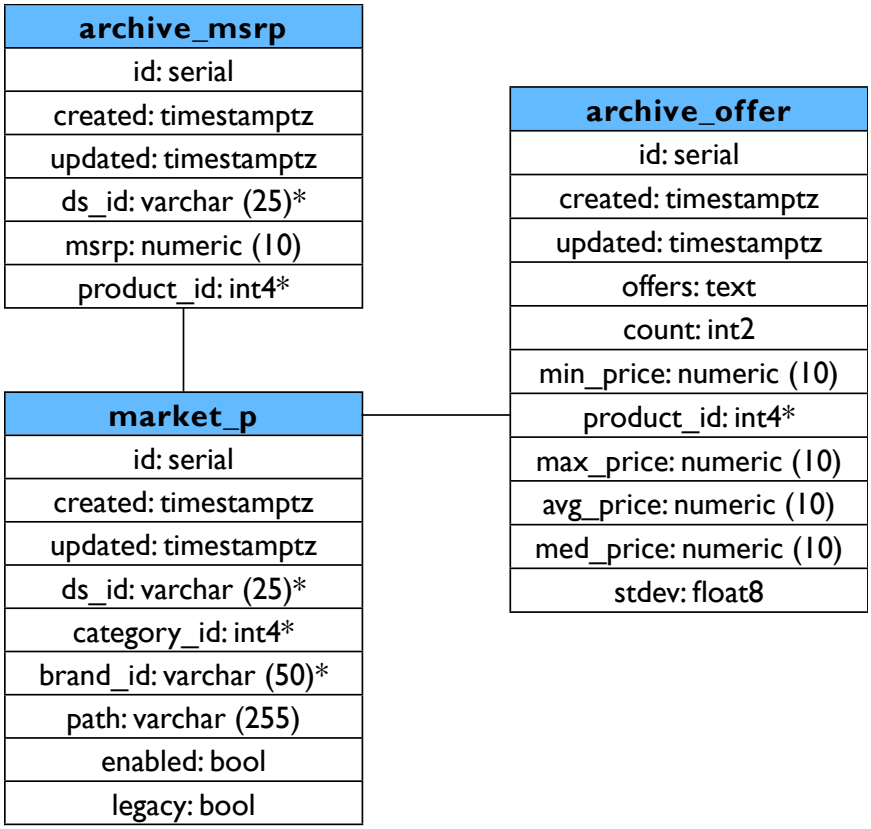
Brands/Manufacturers
Brands are stored in the **market_brand** table and are a full list of each distinct brand name that comes from all of the different feeds. *Aliases* allow a comma separated list of different spellings of each brand to avoid creating a new brand entry for a different way of writing the brand. *Parent* allows hierarchy within Brands if for example the objective is to designate that KitchenAid is actually manufactured by Whirlpool.

Attributes
Attributes are connected to various objects – regions, countries, states, USPS zip codes, feeds, products and categories. Attributes are intended to be generic, with generic object methods.

Internally an attribute has two primary components: key and value. This is far from complete, however, as the attribute is clearly Data as well as must compensate for the value to be normalized over the category.

To accomplish that, all attribute objects have the “N” composite field – containing the *N_value* and the *N_units* elements. When attributes are discovered to be numeric by way of application logic, the “N” is transparently filled out with the *N_value* equal to the normalized value and *N_units* equal to the units part after the first space character.

Archive app



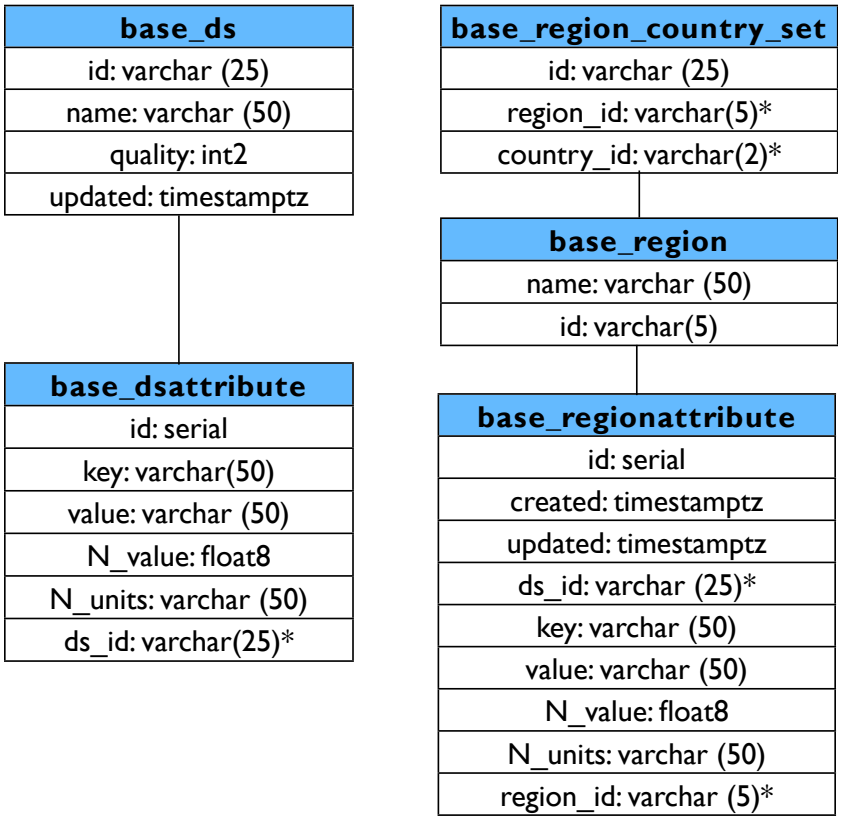
* signifies field is a Foreign Key

The archive app stores historical price information for individual products. It can capture data on a daily frequency.

The **archive_msrp** table stores the manufacturers suggested retail price (*msrp*), which online shopping site it comes from (*ds_id*), and a link to the corresponding product ID (*product_id*).

The **achive_offer** tables stores the minimum (*min_price*), maximum (*max_price*), average (*avg_price*), and median (*med_price*) of all offers from a single day. It also captures the standard deviation of offers (*stdev*) and the number of offers (*count*) for that day. Lastly, the full offers XML (*offers*) is stored to be able to view the details of individual offers.

Base App



* signifies field is a Foreign Key

The base app provides the list of available of data sources and regions that are referenced by other apps in the framework.

The **base_ds** table defines each data source (*name*) which can be either an online shopping API, government data provider, manufacturer/retailer crawled data, etc. The priority score (*quality*) is set from 0 to 100 to determine which data gets used when it is available from more than one data source.

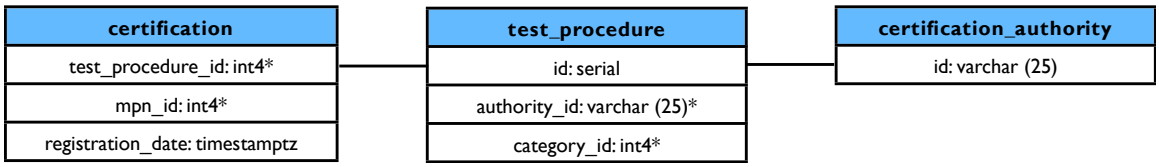
The **base_dsattribute** table allows for the capturing of specific attributes that are relevant to a specific data source. Things such as price, ID, contact name, etc could be stored.

The **base_region** table defines each market (*name*) that would capture product data in the framework. The ID (*id*) links to the **base_region_country_set** table, allowing specification of the country or list of countries (*country_id*) that makes up that market.

The **base_regionattribute** table captures attributes about the region such as the national average electricity rate, % of renewable power supply, and average carbon

intensity of all electric utilities. Each attribute is given a data source (*ds_id*), a variable name (*key*), an attribute value (*value*) that is broken into the numeric part (*N_value*) and units (*N_units*). Each attribute is linked to a single region (*region_id*).

Certification App



*signifies field is a Foreign Key

The certification app captures test procedures, certifying authorities, and product registration dates.

The **certification** table links a product model (*mpn_id*) to a test procedure (*test_procedure_id*) on a specific data (*registration_date*).

The **test_procedure** table stores a list of test procedures per certifying authority (*authority_id*) and category (*category_id*).

The **certification_authority** table contains the full list of certifying authorities (*id*).

Crawled App

crawled_upload
created: timestamptz
updated: timestamptz
ds_id: varchar (25)*
id: varchar (8)
category_id: int4*
src: varchar (100)
enabled: bool

crawled_uploadrule
id: serial
upload_id: varchar (8)*
category_id: int4*
name: varchar(50)
operator: varchar (10)
value: varchar (100)
N_value: float8
N_units: varchar (50)
reject: bool
ignore: bool
rewrite_name: varchar (50)
rewrite_value: varchar (50)
translate_id: int4*

*signifies field is a Foreign Key

The crawled app captures static file data sources and mapping rules used to create products.

The **crawled_upload** table stores a list of all static data sets that are being used to create products. The data source (*ds_id*) must be defined in the base app. Each crawled file is given a system ID (*id*) and can be turned on/off (*enabled*). Crawled files should be assigned to a single product category (*category_id*).

The **crawled_uploadrule** table allows for logical rules to be put in place to normalize data by rewriting attribute names and values. Each upload rule is tied to a single crawled_upload (*upload_id*) and category (*category_id*). It is necessary to input:

- A single attribute (*name*), the logical condition (*operator*) such as "any","in","not","<",">","=","<=",">="
- A single attribute value (*value*) made up of the composite numeric value (*N_value*) and units (*N_units*)
- Whether to reject a product (*reject*) based on having this attribute value. I.e. For Room Air Conditioners one could choose to reject products that have a Type of "Central Air Conditioner"
- Whether to ignore the product attribute (*ignore*) based on having this attribute value. I.e. For Room Air Conditioners one could choose to ignore the Type attribute value of "Air Conditioner" as this does not provide useful information
- Whether to change the name of the attribute (*rewrite_name*) to normalize it across different data feeds.

- Whether to change a specific attribute value (*rewrite_value*) to normalize it across data feeds.
- Whether to move a product containing the specified attribute and value to a different category (*translated_id*). I.e. For Room Air Conditioners one could choose to move products that have a Type of "Central Air Conditioner" to a category for Central Air Conditioners

Datasets App

datasets_upload
created: timestamptz
updated: timestamptz
ds_id: varchar (25)*
id: varchar (8)
category_id: int4*
src: varchar (100)
enabled: bool
alternate_brands: bool
capture_energy: bool
capture_attributes: bool
capture_efficiency: bool
use_direct_match: bool

datasets_uploadstat
id: serial
upload_id: varchar (8)*
updated: timestamptz
total: int4
matches: int4
header: text
format_errors: text
brand_errors: text

*signifies field is a Foreign Key

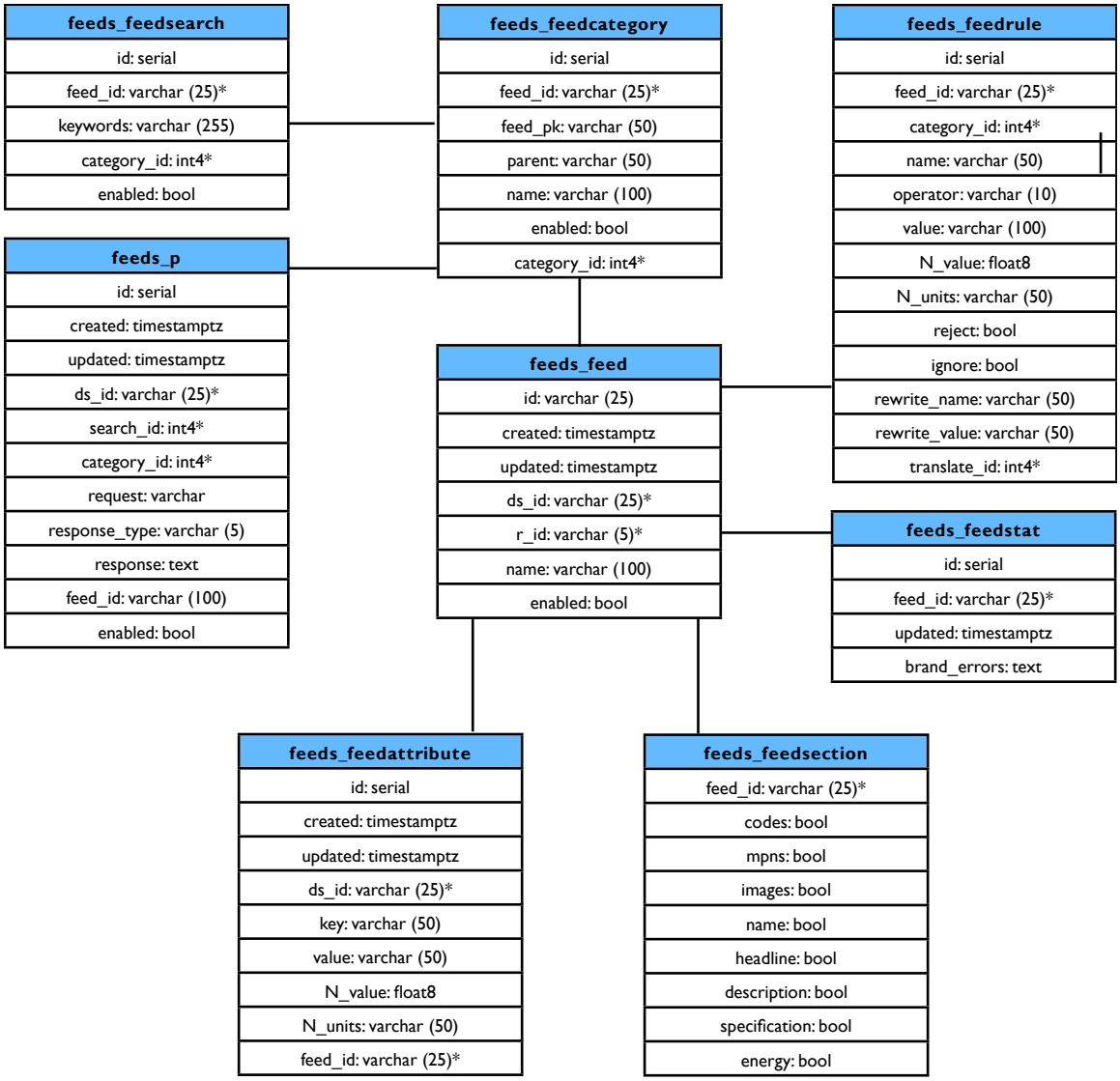
The Datasets App captures the list of certification data sets that will be matched to products.

The **datasets_upload** table has a record for each unique data set, capturing the data source (*ds_id*), category (*category_id*), and allowing for the following parameters:

- *Enabled*: to switch the data set matching on or off. These data sets may be set up to run daily or weekly therefore its necessary to have an ability to enable/disable them.
- *Alternate_brands*: allows for a second brand column to capture manufacturer or other parent relationships
- *Capture_energy*: whether the data set contains operating, standby, or annual energy consumption
- *Capture_attributes*: whether the data set contains product attributes such as size, capacity, color, etc
- *Capture_efficiency*: whether the data set contains information about the product's efficiency such as a rating or certification
- *Use_direct_match*: to determine whether the data set matches directly on the unique product database ID or with logic based on the MPN (model number)

The **datasets_uploadstat** table stores statistics for each run of a **datasets_upload**. Each upload (*upload_id*) is timestamped (*updated*) and provides the total number of product data to match (*total*), number of actual matches with the database (*matches*), a full description of the header row of each file defining what the columns are (*header*), and any errors from formatting (*format_errors*) or unrecognized brands (*brand_errors*).

Feeds App



*signifies field is a Foreign Key

The Feeds App captures details of each online shopping API, its data mapping rules, and the raw product data that is retrieved.

The **feeds_feed** table captures the data source (*ds_id*), region (*r_id*), name (*name*), and whether or not it is live (*enabled*).

The **feeds_feedrule** table allows for logical rules to be put in place to normalize data by rewriting attribute names and values. Each feed rule is tied to a single feeds_feed (*feed_id*) and category (*category_id*). It is necessary to input:

- A single attribute (*name*), the logical condition (*operator*) such as "any","in","not","<",">","=","<=",">="
- A single attribute value (*value*) made up of the composite numeric value (*N_value*) and units (*N_units*)
- Whether to reject a product (*reject*) based on having this attribute value. I.e. For Room Air Conditioners one could choose to reject products that have a Type of "Central Air Conditioner"
- Whether to ignore the product attribute (*ignore*) based on having this attribute value. I.e. For Room Air Conditioners one could choose to ignore the Type attribute value of "Air Conditioner" as this does not provide useful information
- Whether to change the name of the attribute (*rewrite_name*) to normalize it across different data feeds.
- Whether to change a specific attribute value (*rewrite_value*) to normalize it across data feeds.
- Whether to move a product containing the specified attribute and value to a different category (*translated_id*). I.e. For Room Air Conditioners one could choose to move products that have a Type of "Central Air Conditioner" to a category for Central Air Conditioners

The **feeds_feedsearch** table allows for searching of the API using keywords (*keywords*) instead of category IDs to map products to their respective category (*category_id*) in the framework. Each feed's keywords (*feed_id*) can be turned on/off (*enabled*).

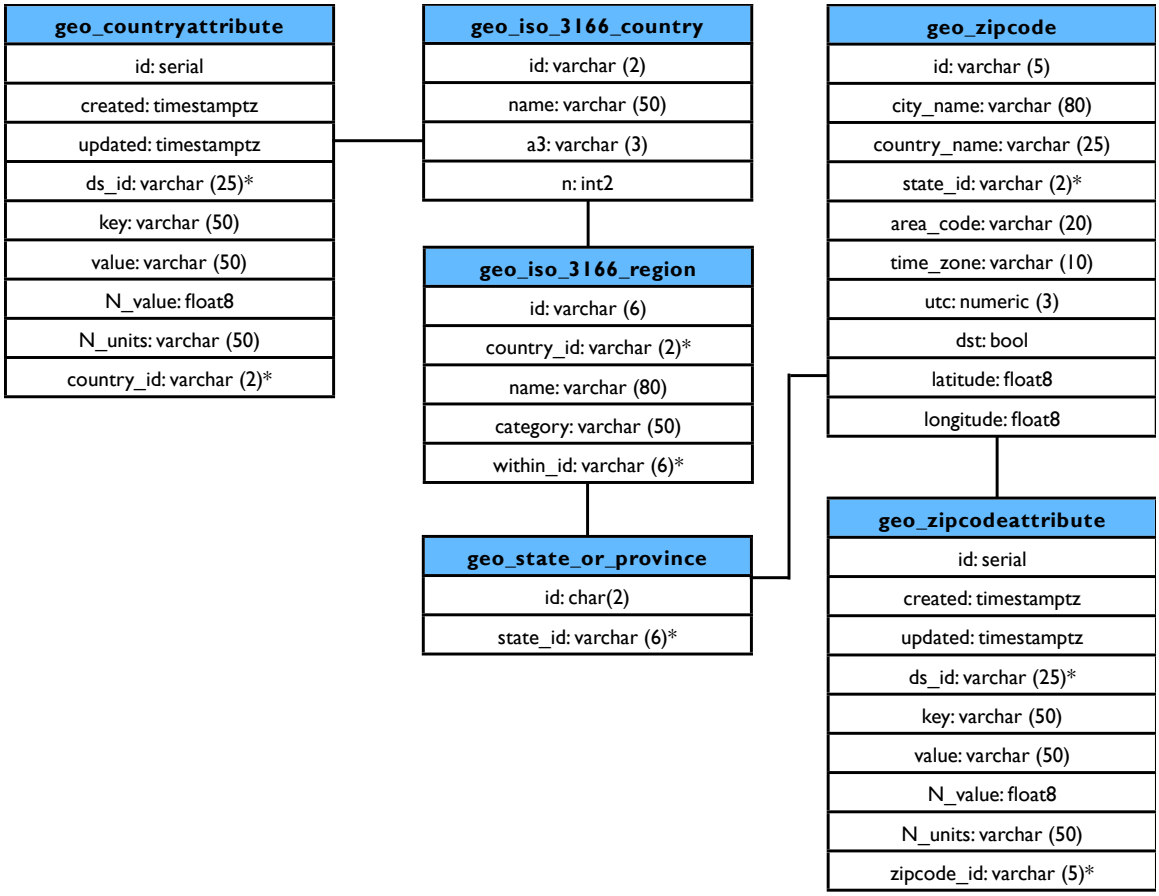
The **feeds_feedcategory** table determines which categories (*name* and *category_id*) by their category IDs from the API (*feed_pk* and *parent*) will be searched for product offers from each feed (*feed_id*).

The **feeds_feedsection** table denotes which parts of the data feed should be captured from the API. UPC/EAN codes (*codes*), model numbers (*mpns*), images (*images*), name (*name*), headline (*headline*), description (*description*), product attributes (*specification*), and energy consumption (*energy*).

The **feeds_feedstat** table captures a log of any brand errors (*brand_errors*) that occur for a feed (*feed_id*) when it is run.

The **feeds_p** table stores raw product data from each of the feeds before any validation logic or feed rules have been applied. This provides a record of exactly what comes back from the feed. The data source (*ds_id*), category (*category_id*), full request (*request*), response (*response*) and response type such as XML or JSON (*response_type*) are always captured. The API feed ID (*feed_id*) is also captured to make it easy to reference this product again.

Geo App



*signifies field is a Foreign Key

The Geo App captures the full list of country, state, and local postal codes available to associate with a market based off the ISO 3166 definition which the International Standard for country codes and codes for their subdivisions.

The **geo_iso_3166_country** table contains a list of country name (*name*), text codes (*a3*), and numeric codes (*n*).

The **geo_iso_3166_region** table contains a list of all subdivisions within a country (*country_id*). Each subdivision (*name*) has a type (*category*) such as province, country, district, etc. It also allows specification of a subdivision (*within_id*) within another subdivision. This allows hierarchy such that a county is within a state that is within a country.

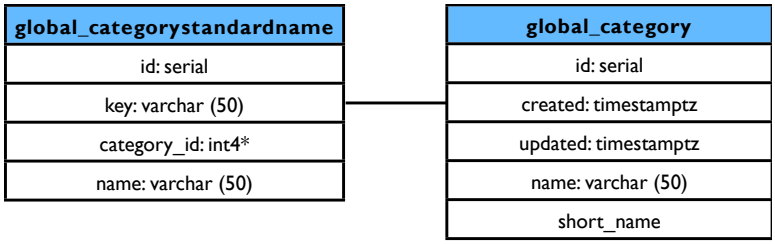
The **geo_state_or_province** table links a state or province to a region (*state_id*).

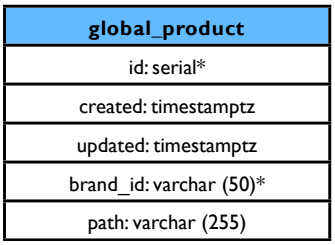
The **geo_zipcode** table stores city (*city_name*), country (*country_name*), state (*state_id*), area code (*area_code*), time zone (*time_zone*), and other location attributes (*utc*, *dst*, *latitude*, and *longitude*).

The **geo_zipcodeattribute** table allows for the capturing of specific key-value attributes related to a specific zip code (*zipcode_id*).

The **geo_countryattribute** table allows for the capturing of specific key-value attributes related to a specific country (*country_id*).

Global App





*signifies field is a Foreign Key

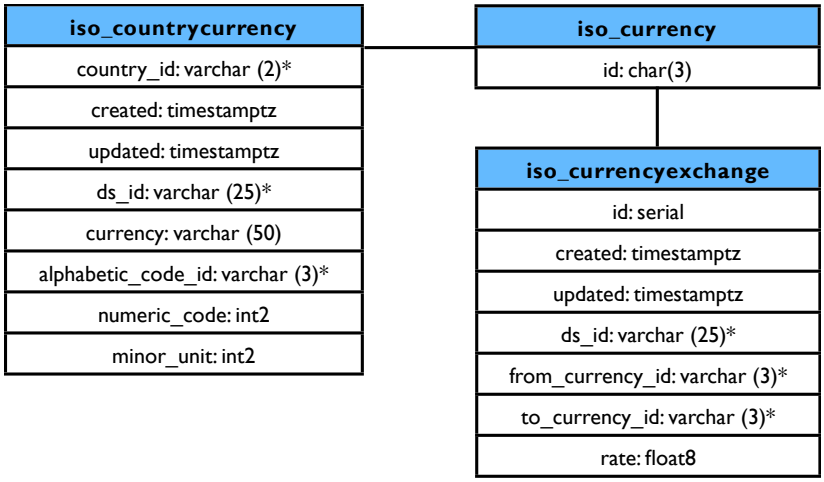
The Global App captures the definition of each category in the global data standard data standard and the list of brands that are recognized internationally.

The **global_category** table captures the category name (*name*) and abbreviated name (*short_name*). It is timestamped (*created* and *updated*).

The **global_categorystandardname** table captures the attributes (*name*) and allowable values (*key*) for each category (*category_id*). This provides a way of using the application layer to enforce the data standard as long as these tables are agreed on and shared across all participating countries.

The **global_product** table captures the list of brands (*brand_id*) that are recognized by the data standard.

Iso App



*signifies field is a Foreign Key

The Iso App captures information about currencies and their exchange rates.

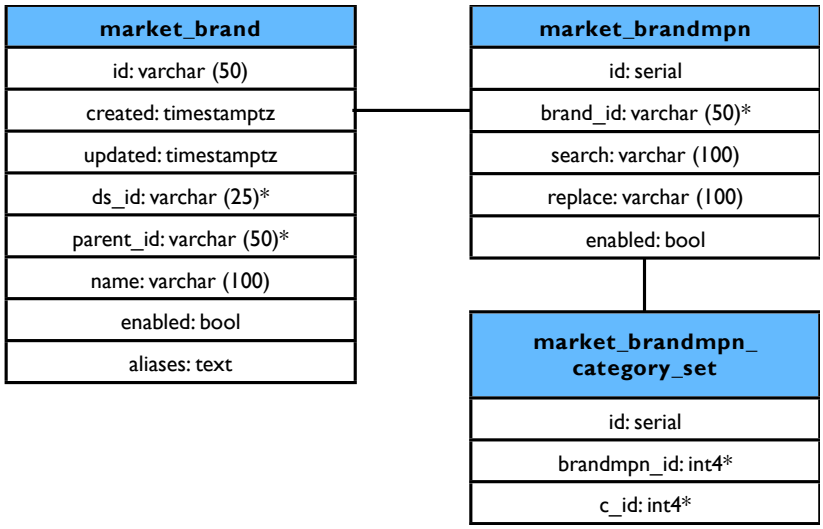
The **iso_currency** table contains a list of all currency abbreviations.

The **iso_countrycurrency** table links a currency code (*alphabetic_code_id*) to a country (*country_id*) with its data source (*ds_id*), currency name (*currency*), numeric code (*numeric_code*), and minor unit (*minor_unit*).

The **iso_currencyexchange** table stores exchange rates between currencies. It contains a timestamp (*created* and *updated*) to allow for updates to occur regularly (i.e. weekly, monthly, quarterly). A standard rate source can be selected (*ds_id*) and then all conversions must be specified from one currency (*from_currency_id*) to another currency (*to_currency_id*) at a specific rate (*rate*).

Market App

Brand



*signifies field is a Foreign Key

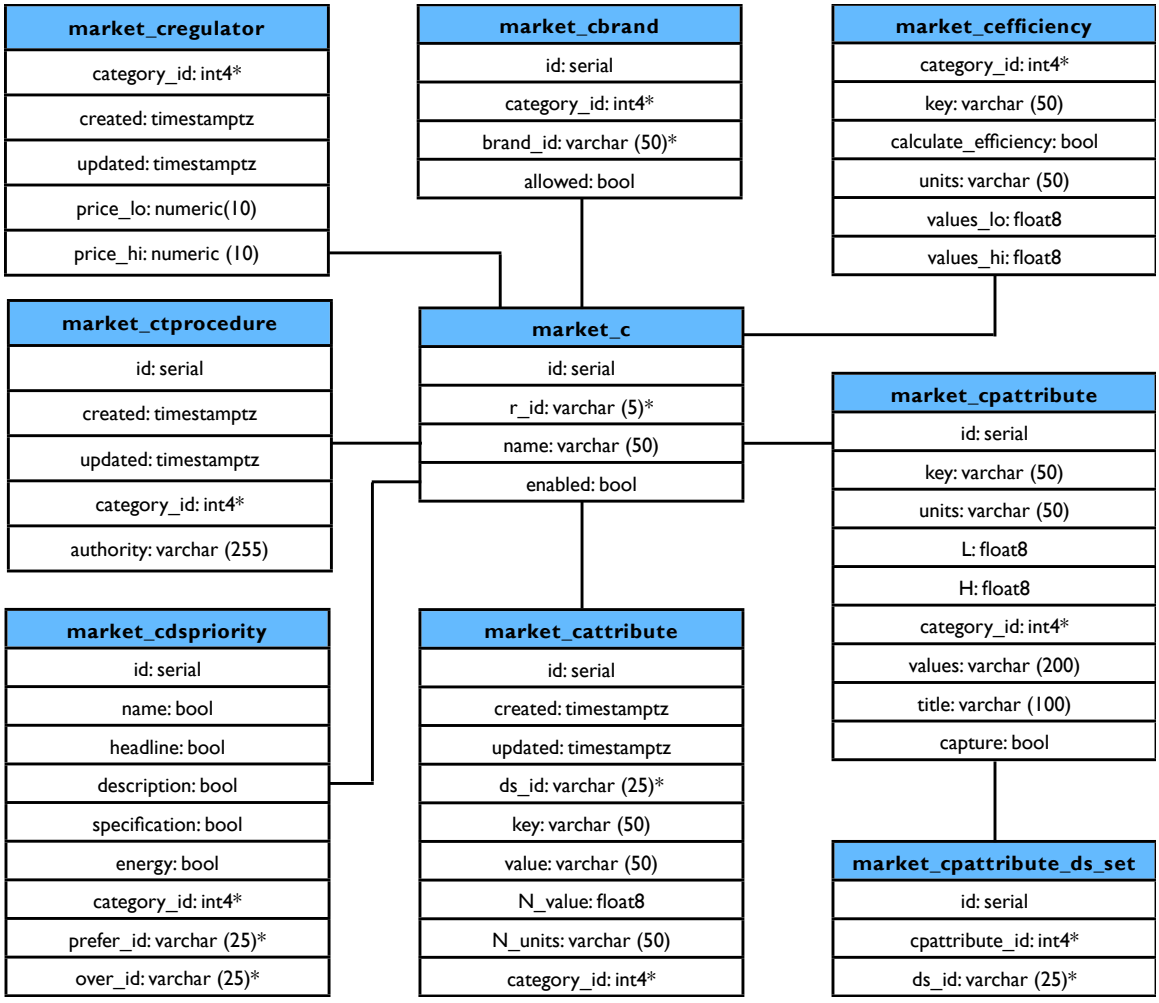
The Brand subsection of the Market App captures the list of brands that are recognized by the system and brand associations by category.

The **market_brand** table is used to create the full list of acceptable brands in the system. Each brand (name) is tagged with a data source (*ds_id*) and any potential variations in how it is written are captured in a comma separated list (*aliases*). Full text searching should be used to resolve these variations. In addition, parent (*parent_id)* relationships can be constructed to capture manufacturers that have multiple consumer brands.

The **market_brandmpn** table allows for rules to be put in place to search (*search*) and replace (*replace*) text within an MPN based on the brand (*brand_id*). If there are extraneous characters such as "-" or "/" these can be removed.

The **market_brandmpn_category_set** table provides a method of allowing/disallowing brands (*brandmpn_id*) for a specific category (*c_id*). This provides a lot of control if certain manufacturer brands should be excluded from a category.

Category



*signifies field is a Foreign Key

The Category subsection of the Market App defines each category, its attributes, and energy and efficiency related ratings.

The **market_c** table contains a each category name (*name*), its region (*r_id*), and whether it is enabled (*enabled*).

The **market_cattribute** table allows for category specific attributes to be defined as key-value pairs. This is used for specific static values that are relevant to only this category.

The **market_cpattribute** table defines the category attributes that will be captured (*capture*). Name (*key*), a comma separated list of units of measurement allowed (*units*), low value if numeric (*L)*, high value if numeric (*H*), category (*category_id*), a comma separated list of allowed values (*values*), and a title for display purposes (*title*).

The **market_cpattribute_ds_set** specifies which data sources each attribute can be populated from.

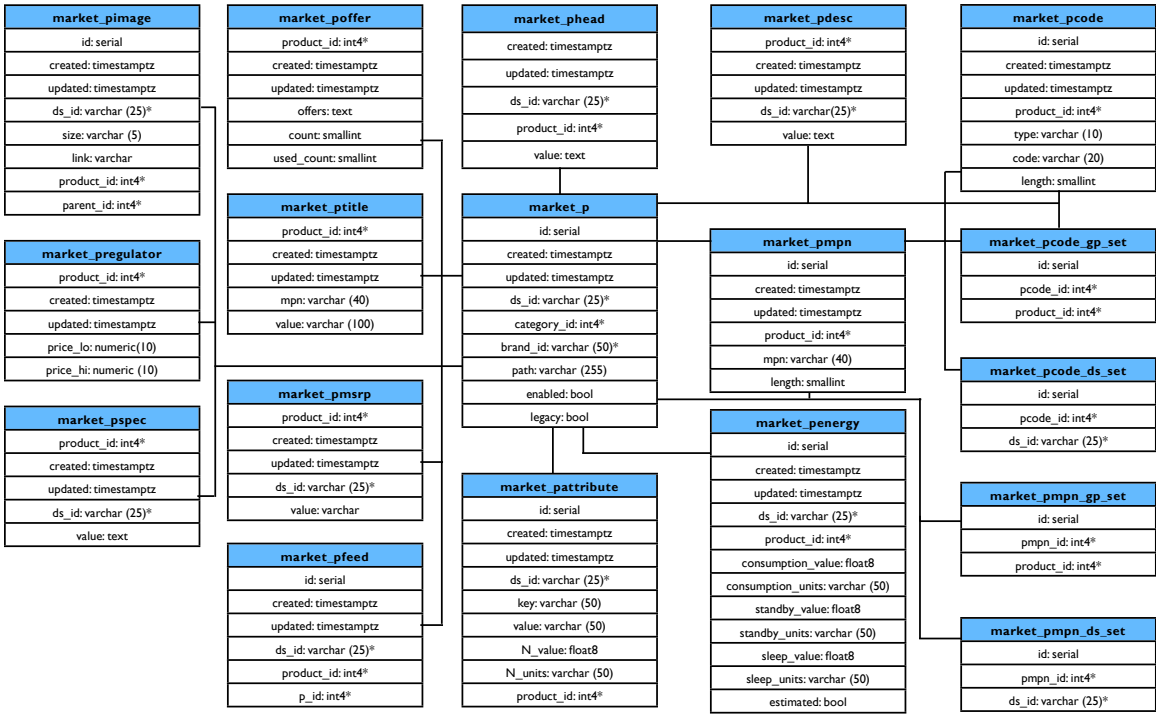
The **market_cefficiency** table specifies how the efficiency of products will be calculated for this category. The key performance attribute (*key*), units of measurement (*units*), and low/high values (*values_lo/values_hi*) are defined to enable the system to do an efficiency calculation per product. An example would be for TVs, the performance attribute would be screen size, units of measurement would be square inches (cm) of screen size, and low/high values could be defined based on the sizes allowed in the market.

The **market_cregulator** table specifies the low and high (*price_lo* and *price_hi*) thresholds for any offer that is allowed into the system. This provides sense checking on any strange data coming in from online shopping APIs.

The **market_cdspriority** table specifies that data source priorities should be overwritten for some product information such as name, headline, description, attributes (*specification*), energy consumption (*energy*). The data source to prefer (*prefer_id*) over another (*over_id*) can be specified which will take precedence over the normal quality levels set in the Base app.

The **market_ctprocedure** table links the category and certifying authority (*authority*) that oversees product certification for that category.

Product



*signifies field is a Foreign Key

The Product subsection of the Market App connects all of the other Apps to create unique products that have data aggregated from a variety of feeds and datatsets. It is the most extensive App in the framework.

The **market_p** table has a record created for each unique product. The data source (*ds_id*), category (*category_id*), brand (*brand_id*), web site address for display on the internet (path), whether the product is off the market (*legacy*), and whether the product is live in the system (*enabled*).

The **market_pattribute** table contains all attributes that are connected to a product (*product_id*). The timestamp (*created* and *updated*), data source (*ds_id*), attribute name (*key*), value (*value* and *N_value, N_units*) are all captured.

The **market_penergy** table contains all energy consumption data related to a product (*product_id*) such as consumption amount (consumption value) and its units such as W or kWh/yr (*consumption_units*), standy consumption (*standby_value* and *standby_units*), sleep consumption (*sleep_value* and *sleep_units*), and whether it is estimated (*estimated*).

The **market_pimage** table contains all links to online images (link) for each product (*product_id*). It specifics the data source (*ds_id*) and size (*size*) to allow for capturing of different image types for different display use cases.

The **market_poffer** table contains the list of all offers for a product (*product_id*). The aggregated offers (*offers*), number of offers (*count*), and number of used offers (*used_count*) are all captured.

The **market_ptitle** table takes the MPN (*mpn*) and makes a product title by combining it with brand (*value*).

The **market_pspec** table contains the full list of all product attributes (*value*) from a specific data source (*ds_id*).

The **market_pmsrp** table contains the manufacturer suggested retail price (value) from a specific data source (*ds_id*).

The **market_pmpn** table contains all of the model number (*mpn*) variations and lengths (*length*) for a product. It is possible to capture more than one if manufacturers and retailers refer to a product in different ways.

The **market_pmpn_ds_set** table associated each product mpn (*pmpn_id*) with its respective data source (*ds_id*).

The **market_pmpn_gp_set** table associates a product (*product_id*) with MPNs in another market (*pmpn_id*) for cross market mapping. For example a Samsung TV in the US could be sold under the UNEH4000 MPN and UAEH4000 MPN in Australia.

The **market_phead** table contains the product headline (*value*) that is a longer, one line title from a specific data source (*ds_id*).

The **market_pdesc** table contains the full product description (*value*) which is a paragraph providing an overview of key features for a product.

The **market_pcode** table contains all of the associated GTIN codes (*code*) for a product such as an ASIN, UPC, EAN (*type*). The length of each code (*length*) is stored.

The **market_pcode_ds_set** table associates each product code (*pcode_id*) with its respective data source (*ds_id*).

The **market_pcode_gp_set** associates the a product (*product_id*) with additional product codes (*pcode_id*) in other markets.

Tracking App

tracking_sales
id: serial
created: timestamptz
updated: timestamptz
mpn_id: int4*
code_id: int4*
units_shipped

tracking_upload
id: varchar(8)
created: timestamptz
updated: timestamptz
ds_id: varchar (25)*
category_id: int4*
src: varchar (100)
enabled: bool

*signifies field is a Foreign Key

The Tracking App allows for capturing of sales data at a model number level into the framework.

The **tracking_sales** table captures the number of units sold (*units_shipped*) over a time period (*code_id*) by product mpn (*mpn_id*).

The **tracking_upload** table allows for a static file to be uploaded to match the sales data with mpns. Uploads must be processed one at a time for a single category (*category_id*) and data source (*ds_id*).