Engineer

Star schema

★ a layout with one fact table and dimension tables around it

What it is

A star schema is a way of laying out tables for analysis. One table in the middle records business events, such as sales, shipments or payments, as numbers. Around it sit the tables that describe those events: which product, which store, which day, which customer. Drawn as a diagram, with a line from the middle table to each of the others, it looks like a star.

The table in the middle is the fact table, and the tables around it are dimension tables. The Kimball Group's description is short: "Star schemas characteristically consist of fact tables linked to associated dimension tables". The link is an ordinary key. Each dimension table has a single primary key, and each fact row carries that key as a foreign key, one for every dimension that describes the event.

The split follows a distinction the Kimball Group draws between measurement and context. A dimensional model is built around the measurement events of a business process: an item scanned at a till, an order shipped, a claim paid. The numbers the event produces, such as the quantity and the amount, are the facts. Everything that says who, what, where, when, why and how is context, and it lives in the dimensions.

The two kinds of table have opposite shapes. At the most detailed level, a fact table holds one row per event, with the numeric measures, a foreign key to every dimension, and sometimes a reference such as an invoice or receipt number. That number is all that is left of the document once its date, customer and store have become dimensions, so it is stored on the fact row with no dimension table behind it, and the Kimball Group calls it a degenerate dimension. A dimension table, by contrast, is wide, flat and deliberately denormalised, with many

descriptive text columns. A product dimension repeats the brand, category and department on every product row instead of splitting them out into tables of their own.

A star schema is the relational form of a dimensional model, built from ordinary tables in an ordinary SQL database. The other form the Kimball Group describes is the OLAP cube, a multidimensional database holding the same attributes and facts, queried in languages with more analytic power than SQL, such as MDX, with XMLA providing a protocol for access. Cubes are often built from a star schema, as a last step before users see the data or as a layer of aggregates. The modelling rules are the same for both, and the Kimball Group calls them architecture-neutral: they apply to a Kimball-style data warehouse and to Bill Inmon's hub-and-spoke Corporate Information Factory alike.

Use the concept in a decision

Put "What is a star schema?" to work in a real decision. We can help relate the concept to the system you are building or buying. Sales, operations and finance figures brought together, so two departments stop bringing two answers to the same meeting.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Made a geographical data pipeline 50x faster — 50x faster geographical data pipelines

Made a geographical data pipeline 50x faster Get in touch →

Why it matters

The shape makes analytical questions look alike. Pick the facts to add up, join the fact table to the dimensions that describe them, filter on some dimension columns and group by others. The Kimball Group describes fact tables as the main target of calculations and dynamic aggregations, and dimension attributes as the main source of query constraints, groupings and report labels. An analyst who has learned that pattern once can write the next query against a new fact table without help, and a reporting tool can build it from a list of fields, because every star has the same layout.

Readability is the second reason. An operational database is designed to store each fact once and keep it consistent, so a product's category usually lives in a separate lookup table, joined back by an id. That is the right design for the system that takes orders and the wrong one for someone looking for sales by category, who first has to work out which tables to join and in which order. A flat dimension puts every descriptive column in one place, written out in words. The

Kimball Group warns against the alternative, normalising a dimension's hierarchy into further tables, because business users find the result hard to understand and navigate.

The third reason is that a well-built star can answer questions nobody asked when it was designed. The grain of a fact table, the exact meaning of one row, is declared before any dimension or fact, and the Kimball Group recommends starting at the atomic grain: the lowest level the business process captures. A table with one row per receipt line can be summed by day, store, brand or any combination of them. A table that keeps only daily totals per store can never be broken down by product, however the question is phrased. In the Kimball Group's terms, atomic data withstands unpredictable queries, while rolled-up data presupposes the questions it will be asked.

The price is redundancy on purpose. The department name is repeated on every product row, and the load that builds the star has to keep those repetitions consistent and decide what happens when a description changes. The star does not replace the operational model either; it is built from it by a load process, and it is only as current as that load's last run.

What it looks like in practice

A retailer wants sales by product category and month. The business process is the till receipt, and the grain is declared as one row per receipt line. The product dimension is one flat table, the fact table holds a key for each dimension, the receipt number as a degenerate dimension and two measures.

```
CREATE TABLE dim_product (
    product_key  bigint PRIMARY KEY,   -- surrogate key, owned by the warehouse
    sku          text NOT NULL,        -- the code in the source system
    product_name text NOT NULL,
    brand        text NOT NULL,
    category     text NOT NULL,
    department   text NOT NULL
);

CREATE TABLE fact_sales (
    date_key       integer NOT NULL REFERENCES dim_date,
    store_key      integer NOT NULL REFERENCES dim_store,
    product_key    bigint  NOT NULL REFERENCES dim_product,
    receipt_number text    NOT NULL,   -- degenerate dimension
    quantity       integer NOT NULL,
    sales_amount   numeric(12, 2) NOT NULL
```

```
);

SELECT d.year_month, p.category, sum(f.sales_amount) AS sales
FROM fact_sales AS f
JOIN dim_date AS d USING (date_key)
JOIN dim_product AS p USING (product_key)
WHERE p.department = 'Grocery'
GROUP BY d.year_month, p.category
ORDER BY d.year_month, p.category;
```

The query has the shape every star query has: the fact table, one join per dimension, filters and groupings on dimension text, and a sum over a fact. Adding the store's region to the report is one more join and one more column in the `GROUP BY`; the fact table does not change. The receipt number stays on the fact row, so the lines of one basket can still be counted together, but it gets no dimension table, because there is nothing more to say about it.

A star is one answer to the promise of the database design and modelling service: a structure that matches how the business actually works, so the questions needed later can still be asked. The operational model it is built from is the other half of that work. In current work for NextMariner, the founder modelled a maritime domain in 348 normalised tables across 34 PostgreSQL schemas, covering professionals, companies, ships, jobs and reviews. There, small stable lists such as statuses, types and categories became SMALLINT lookup tables, which keeps rows compact and joins cheap, and real foreign keys and constraints make integrity the database's job. That is the right shape for a system that writes rows one at a time and must stay correct. A star built for reporting on top of such a system goes the other way: it turns the codes back into words in flat dimensions, because its reader is a person rather than the application.

Where it goes wrong

The dimensions are normalised, and the star turns into a snowflake. Splitting brand, category and department out of the product dimension into tables of their own looks tidy to anyone used to operational design. The Kimball Group calls the result a snowflake and advises against it: business users find it hard to understand and navigate, and the extra joins can slow queries down, while a flat dimension holds exactly the same information. Keep each dimension as one wide table and let the load repeat the hierarchy on every row.

The grain is set to fit the first report. A fact table of daily sales per store answers the dashboard it was built for and nothing below it; the first question about a single product or basket needs a new load from the source. A table that mixes daily totals with individual lines is worse, because any sum over it counts the same sale twice. Declare the grain first, start at the atomic level the process records, and give each grain a fact table of its own. Summary tables can then be added on top for speed, as separate tables.

The dimension attributes are the source system's codes. Operational systems store compact codes, and copying them straight into a dimension produces reports whose row labels only the source system understands. The Kimball Group allows codes as attributes but says the most useful attributes are verbose descriptions, since those are what end up as labels on a report. Decode every code into words during the load, and keep the code in a column beside the description for anyone who has to trace a row back to its source.

The fact table is shaped like a report. It is tempting to design it from the spreadsheet the business already uses, with a column per month or a row per line of the management report. The Kimball Group is explicit that a fact table's design rests on the physical activity being measured, not on the reports that will be drawn from it. Report layouts change; the till receipt does not. Model the event, one row per occurrence at the declared grain, and let queries produce the layouts.

Further reading

- Star Schemas and OLAP Cubes — the definition, facts and dimensions as measurement and context, and how cubes relate to stars.
- Grain — declaring what one fact row means, and why the atomic grain comes first.
- Fact Table Structure — numeric measures and foreign keys on one row per event, designed from the activity rather than the report.
- Dimension Table Structure — wide, flat, denormalised tables whose descriptive attributes drive filters, groupings and labels.
- Snowflaked Dimensions — why normalising a dimension's hierarchy makes it harder to use.
- Microsoft: Multidimensional model data access — MDX as a query and calculation language, and XMLA as the interface through which a client can send an MDX statement.

Glossary

Choose for your actual constraints

Bring a representative dataset, the questions it must answer and the current bottleneck. We can agree how to measure quality, throughput and recovery.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is a star schema?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/star-schema/