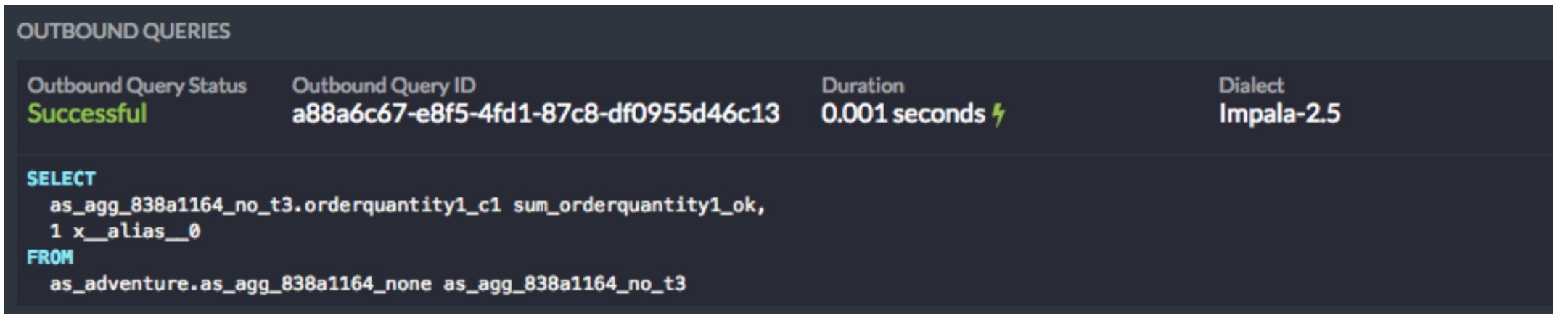
About Query Cache

The AtScale engine is enabled by default to cache query results locally on an AtScale node. When subsequent, identical queries run, the AtScale engine can direct them to cached results, improving query performance. However, caching is opportunistic and not guaranteed, as a number of factors determine whether the cache is used for any particular query.

The **Queries** page in the Design Center indicates when an outbound query (that is, a query rewritten by the AtScale engine to match the flavor of SQL that is understood by an SQL-on-Hadoop engine or Google BigQuery) has run against the query cache. The indicator is a lightning bolt that appears next to the reported duration of the query, as in this example:

Figure 1. The lightning-bolt indicator next to the reported duration of a query



When this indicator appears, it is either green or yellow, according to these rules:

- The duration of any subquery that hits the cache is accompanied by a green lightning bolt.
- The duration of any query for which all subqueries hit the cache is accompanied by a green lightning bolt.
- The duration of any query for which a subset of subqueries hits the cache is accompanied by a yellow lightning bolt.

When you are on the **Queries** page, you can mouse over the lightning bolt for an explanation of its color.

- In AtScale 7.0.0 and later, caching takes place for all users when impersonation or delegation is enabled, unless the settting authorization.query.canary.always is disabled.
- When a query does run against the query cache, there might be no record of the data access outside of AtScale's own records. If you are relying on other systems for auditing and logging, you should disable the query cache.
- The AtScale engine uses the query cache opportunistically. There is no guarantee that any particular query will be run against the query cache.
- The AtScale engine can remove cached results at any time as a result of memory usage, usage statistics, changes to projects, aggregate definitions, or other configuration settings.

- In an AtScale cluster, each node has its own query cache. Cached query results are not shared among nodes.
- Caching is enabled by default. When the cache is configured with the default values for its configuration settings, its total memory footprint will be less than 1 GB of RAM in most cases.

There are three settings for configuring the query cache. In all but a few cases, the default settings will make the best use of the cache.

- Two of the settings control the size of the query cache: query.cache.entry.sizeLimit and query.cache.eviction.sizeLimit. The size of the cache is measured in cells. The number of cells is the number of dimensional member combinations times the number of keys and attributes required to process the query.

  For example, take a query that asks for ad clicks and impressions by state, where State has both a name and a key. If we are restricted to the United States, this query might produce one member for each of the fifty states, each of which has four atoms of data: clicks, impressions, state_key, and state_name. The total number of cells of data would amount to roughly 200. (In some cases, the number might be smaller due to sparsity.)

  Important: The number of cells of data involved in a query's results is not the same as the number of XMLA cells in the results of a multidimensional query.

  The setting query.cache.entry.sizeLimit is the maximum number of cells to allow in a single entry in the cache. In most cases, you will not need to increase the default value of this setting. The query cache is designed for frequent queries that have relatively small result sets. Increase this value only if you have clear evidence of frequently repeated queries that result in more cells than the default limit. This value should not exceed 1% of the value for query.cache.eviction.sizeLimit.

  Remember: To disable caching, set the value of query.cache.entry.sizeLimit to zero.

  The setting query.cache.eviction.sizeLimit is the maximum number of cells to allow in the entire cache. If you have a wide variety of frequently repeated queries, you might benefit from raising this limit. Expect each additional cell to have a memory cost of about two hundred bytes.

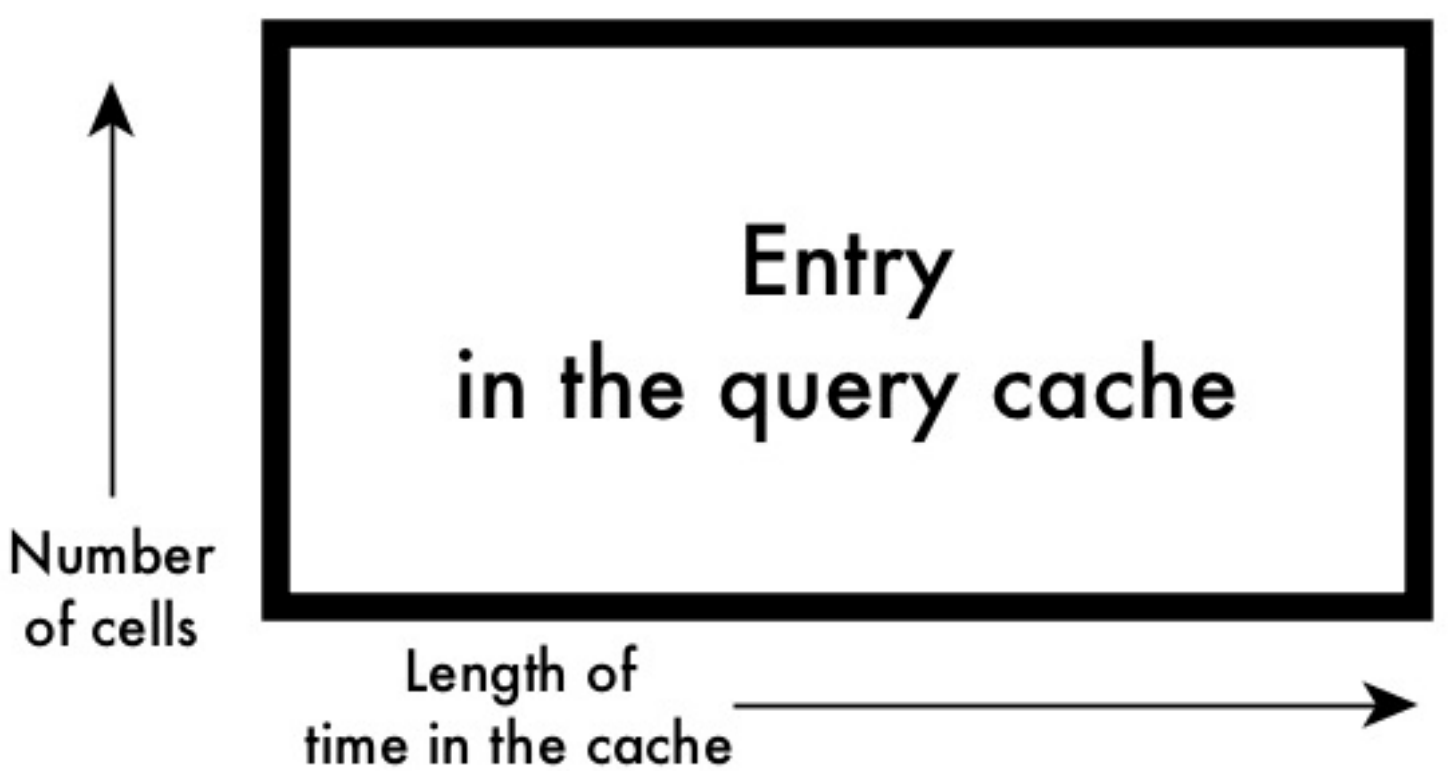
- The setting query.cache.eviction.entryAge is the maximum time that data can remain in the cache. Entries might be removed before they remain this long. Moreover, the cache is always cleared when aggregate instances are rebuilt and when projects are republished. The value is in the format x.minutes, y.seconds, z.hours. For example, a limit of four hours, 1 minute, and 30 seconds is 1.minute, 30.seconds, 4.hours.

  If you do not want to accept the default value, set the value to the maximum amount of staleness you want to allow in the cached data, relative to the freshness of the data that is in the fact tables and aggregate instances that are being queried.

How These Settings Work Together

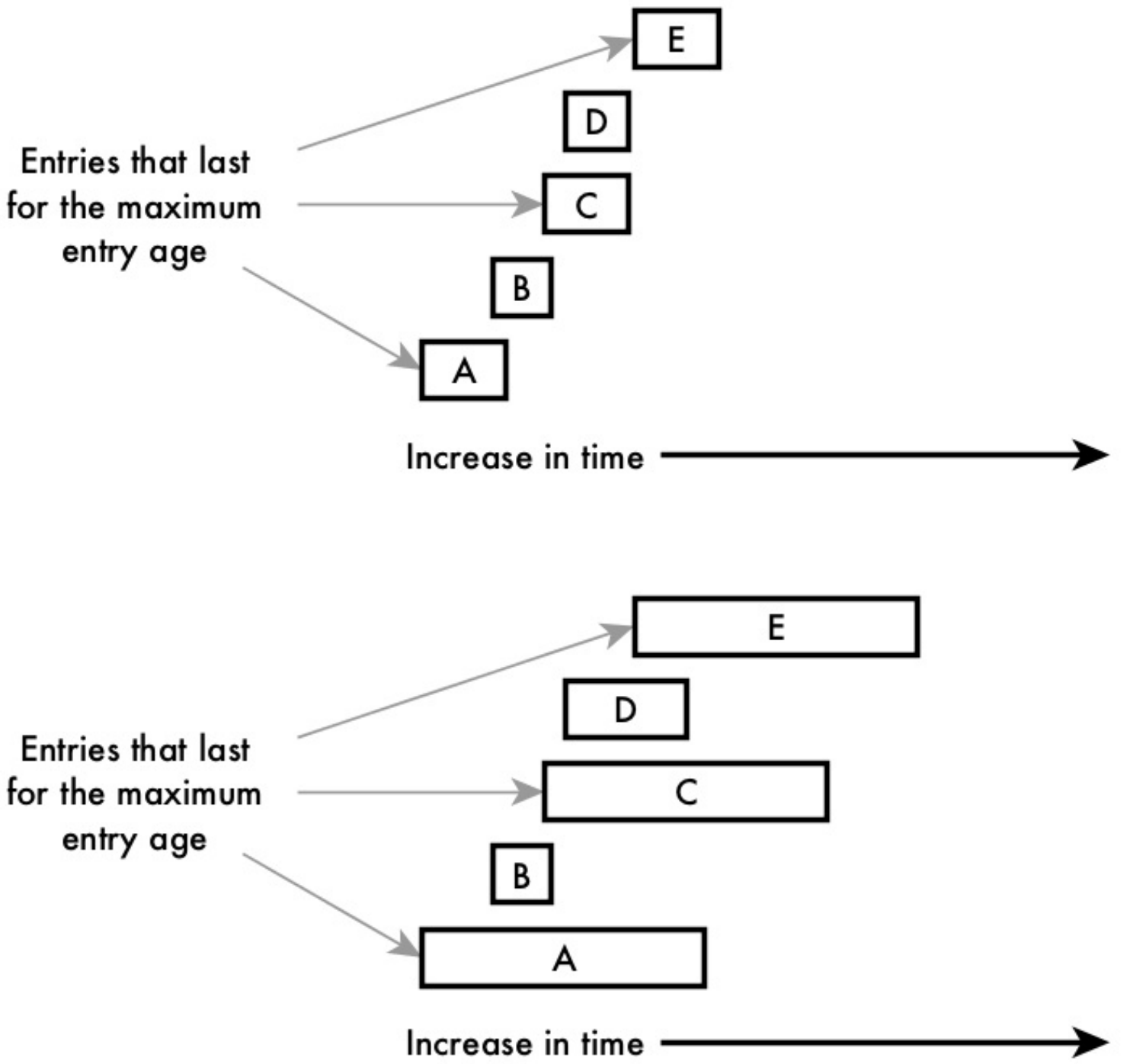
This section describes the three settings by presenting diagrams that abstract and simplify to a high degree how the settings are used. In each diagram, rectangles represent entries in the query cache. The vertical axis represents the number of cells in an entry, and the horizontal axis represents the length of time that an entry remains in the query cache.

Figure 2. A representation of an entry in the query cache



The number of cells in an entry is approximately the number of attributes times the number of values produced.

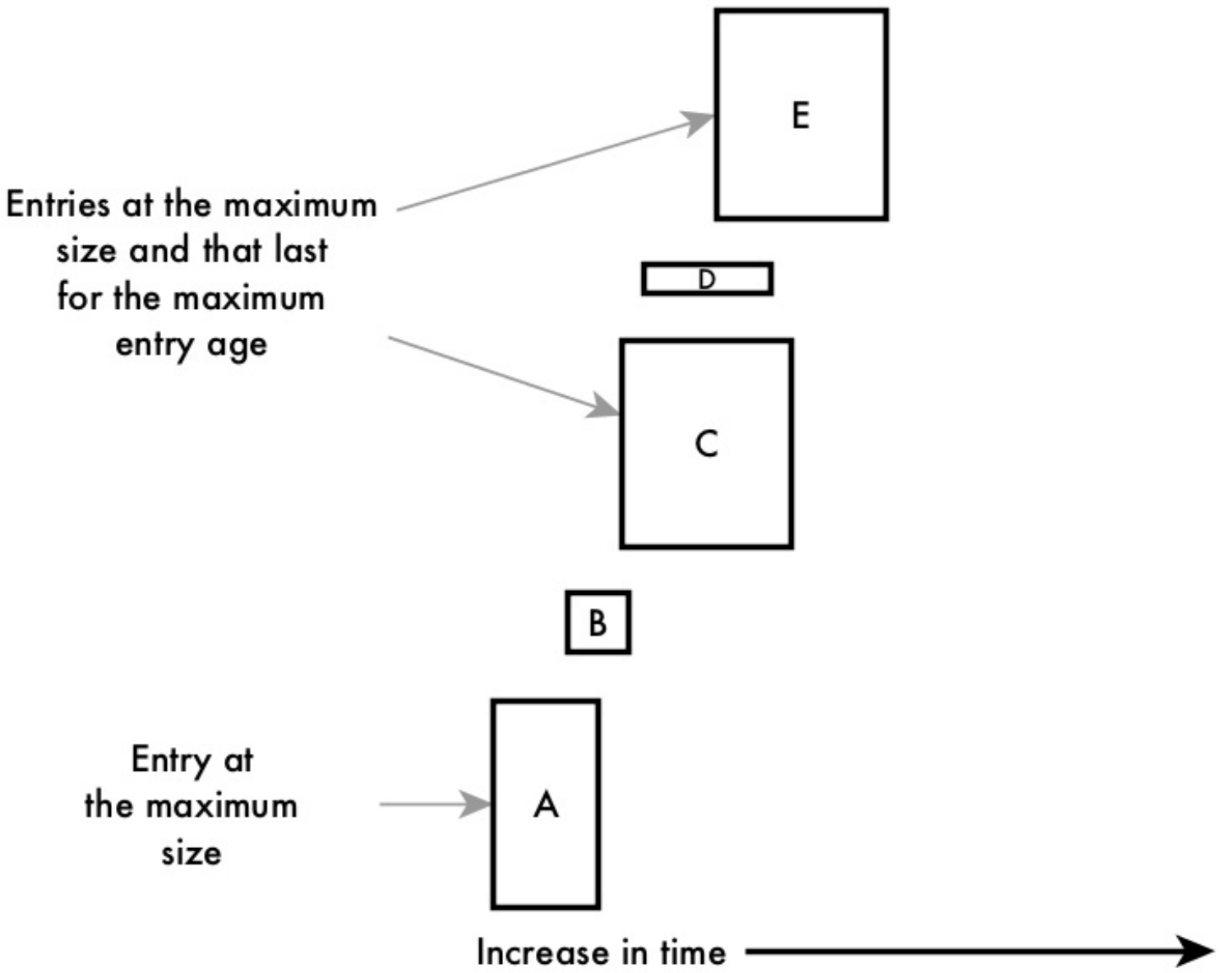
The next two diagrams illustrate the setting `query.cache.eviction.entryAge`. Both diagrams show the same number of entries. To simplify the diagrams, every entry has the same number of cells, and therefore is of the same height as the other entries. The entries appear in the order in which they are created in the cache.

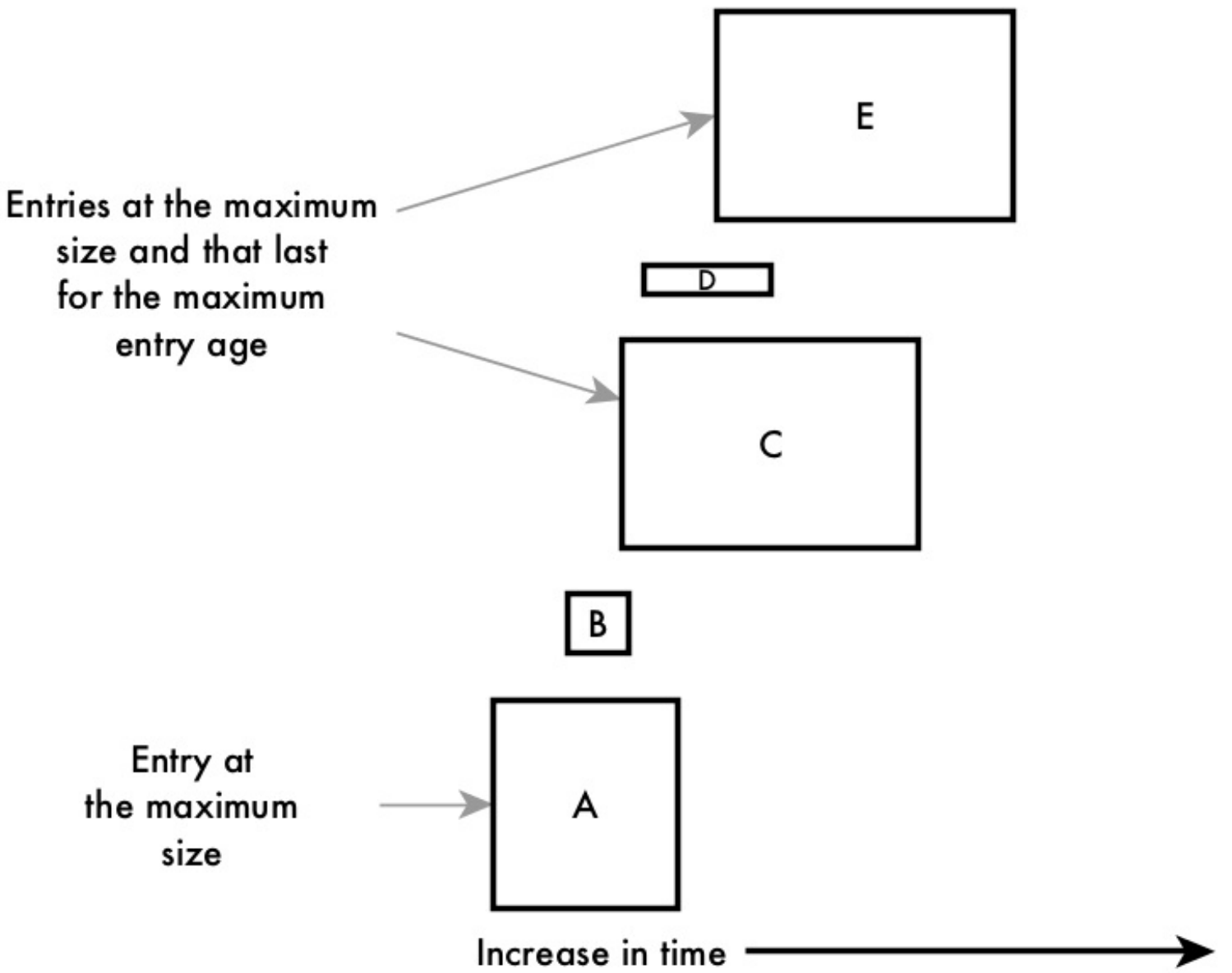


In the top diagram, entries A, C, and E reach the maximum entry age. In the bottom diagram, the maximum entry age is set for a longer duration.

If the entry size limit is set to 0, there will be no entries in the cache.

The next two diagrams illustrate the interaction of entry age with entry size limit. Both diagrams show the same number of entries as the last two diagrams did. This time, however, the number of cells per entry is not uniform. In the top diagram, entries C and E reach the maximum entry age; in the bottom diagram, the maximum entry age is set for a longer duration.


E
Entries at the maximum size and that last for the maximum entry age
D
C
B
Entry at the maximum size
A
Increase in time


E
Entries at the maximum size and that last for the maximum entry age
D
C
B
Entry at the maximum size
A
Increase in time

The last diagram shows greatly simplified interactions between entry age, entry size limit, and eviction size limit. The diagram is a chart that shows the total number of cells in the query cache along the vertical axis and arbitrary, equal units of time along the horizontal axis. All rectangles of a single color constitute a single entry. The maximum entry age is set to be five of these units, as shown by entry C.

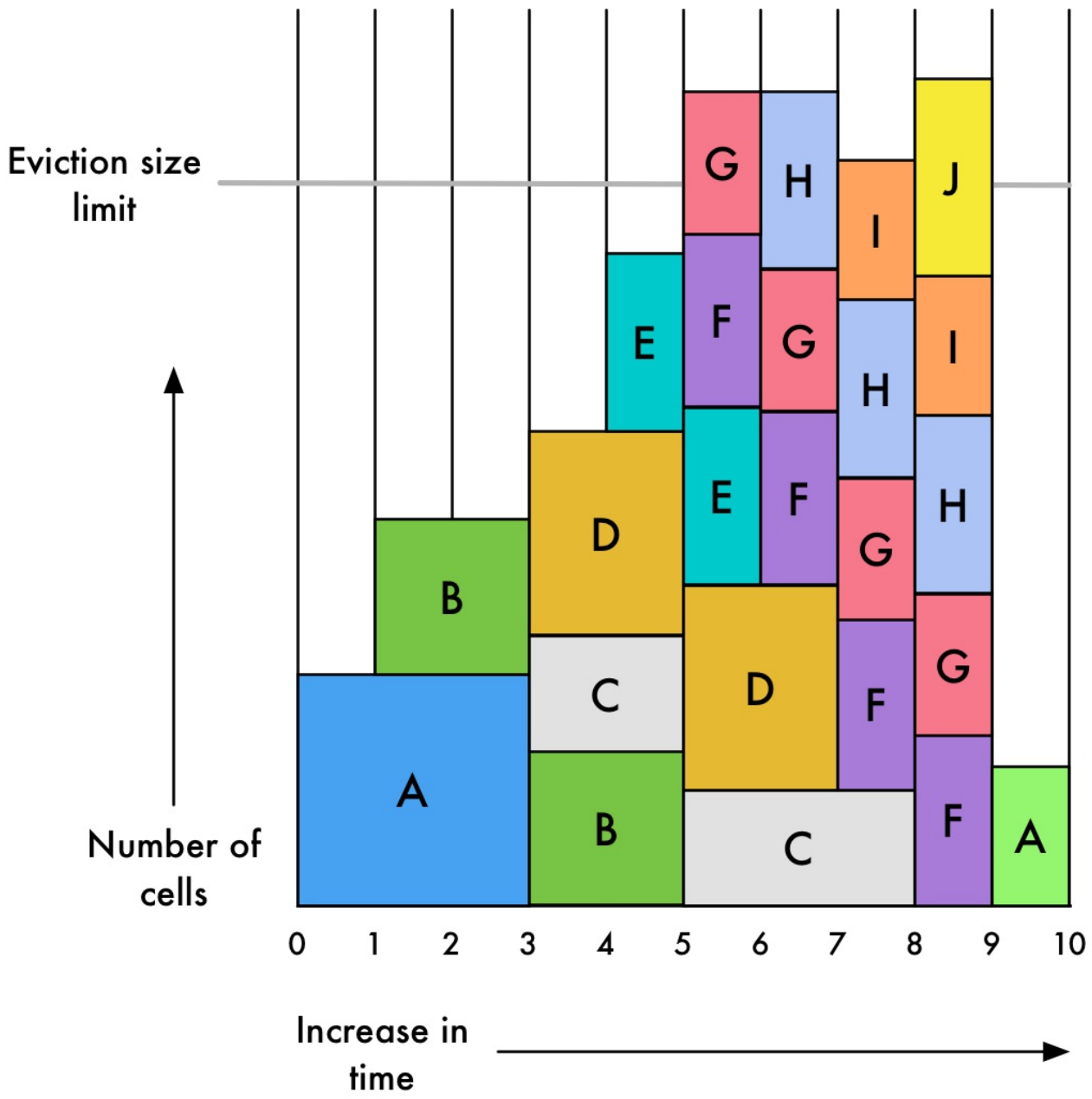


Table 1. The scenario depicted in the diagram

Unit of Time	Description
1	Entry A is added to the cache. It contains the maximum number of cells allowed per entry.
2	Entry B is added to the cache
3	Both entries A and B continue to be the only entries. However, at the end of this unit of time, entry A is removed before it reaches the age limit. The AtScale engine uses several criteria to determine whether it should remove an unexpired

	entry. For example, the engine might determine that the entry is not useful enough to continue in the cache.
4	Entries C and D are added.
5	Entry E is added.
6	Entry B is removed before it reaches the age limit. Entries F and G are added, bringing the total number of cells in the cache past the eviction size limit.
7	Entry E is removed before it reaches the age limit, as part of the pruning of the cache that the engine carries out when the eviction size limit is exceeded. However, entry H is added, again raising the total number of cells higher than the eviction size limit.
8	Entry D is removed early, also as part of the pruning process. Entry I is added. Entry C expires, having reached the age limit. When it expires, it brings the number of cells below the eviction size limit, so the AtScale engine does not need to prune any entries.
9	Entry J is added, the cache exceeding the eviction size limit again as a result.
10	AtScale engine clears the query cache and initiates a rebuild of aggregate-table instances. Immediately after the cache is cleared, it is available for new entries. The first new entry is added.

Remember: The presence of a result set in the cache is not a guarantee that the AtScale engine will use it to satisfy queries that are identical to the query that produced the result set. This last diagram shows only the content of a simplified cache, and it does not show the dynamic interactions between the cache, the AtScale engine, the aggregate tables, the physical fact tables in the data warehouse, and the queries issued by users of client BI applications.