TCGBERG · ANALYTICS METHODOLOGY

Index Engine Methodology

Bloomberg-style indices over trading card atoms: divisor-adjusted, market-cap weighted with confidence dampening.

Version 1.0 | May 2026
Status: Specification — ready for implementation
Author: CK
Scope: 25 auto-derived indices for Pokémon Base Set
Depends on: SCRUM-61 (Fair Value) + SCRUM-62 (Market Cap)

PROBLEM STATEMENT

1. From Atoms To Indices

Fair value tells us what one copy is worth. Market cap tells us what all copies are worth. But neither answers the question every serious investor wants: *how is the market moving?* For that, we need a time series that tracks the aggregate behavior of a basket of constituents — an index.

Without indices, comparing performance across cards, print runs, or grade tiers requires ad-hoc charts and inconsistent baselines. Indices give us canonical reference series — 'Pokémon Base Set PSA 9 is up 23% YTD' becomes a meaningful, defensible claim.

The engine produces 25 auto-derived indices for Pokémon Base Set in v1, plus all the machinery to add indices for future sets without manual curation.

TAXONOMY

2. Two Scope Levels, Five Grade Variants

Indices are auto-derived from the data structure rather than manually curated. Each index is uniquely defined by:

- **Scope** — either a set (e.g., Pokémon Base Set) or a print run within a set (e.g., 1st Edition Shadowless across all Base Set cards)
- **Grade variant** — composite (PSA 7-10 weighted), or individual grade (PSA 10, PSA 9, PSA 8, PSA 7)

Print runs are discovered systematically by grouping printings within a set on attribute flags (`is_first_edition`, `is_shadowless`, `is_unlimited`, etc.). No hardcoded set-specific logic. The same method generates indices for Jungle, Fossil, or any future set added to the database.

For Pokémon Base Set, the auto-discovered print runs are:

Print Run	Filter	Cards	Printings
Unlimited	is_unlimited=T, variant='Unlimited'	102	102
1999-2000 Copyright	is_unlimited=T, variant='1999-2000 Copyright'	102	102
Shadowless	is_first_edition=F, is_shadowless=T	102	102
1st Edition Shadowless	is_first_edition=T, is_shadowless=T	102	102

Four print runs × 5 grade variants + 1 set × 5 grade variants = **25 indices total**.

ELIGIBILITY

3. Constituent Eligibility

An atom (a specific printing × PSA grade combination) is eligible to be a constituent if:

- It has a non-NULL `fair_value` for the target date

- Its `population` is non-zero (latest known snapshot, regardless of date)
- It matches the index's scope filter (print run or set) and grade filter

Confidence score is **not** used as an eligibility filter. Low-confidence atoms are included but down-weighted (see Section 5). This avoids constituent churn from temporary low confidence and lets every legitimate atom contribute proportionally to its quality.

An index publishes only if it has at least **5 eligible constituents**. Indices with fewer constituents are not computed (deferred until breadth grows). This avoids one-or-two-atom indices that are statistically meaningless.

When an atom loses eligibility (loses fair_value or population), it is removed from the index after **1 consecutive month of ineligibility** at the next monthly rebalance. A single date's missing data does not cause removal — only sustained loss does.

CORE MATH

4. Divisor-Adjusted Index Value

The fundamental insight: **population growth should not inflate the index**. Only price action should move the index level. Population changes must be absorbed into a divisor that keeps the index honest.

```
index_value(t) = aggregate_effective_market_cap(t) / divisor(t)
```

Where `aggregate_effective_market_cap` is the sum of confidence-weighted market caps across all constituents (Section 5), and `divisor` is maintained so that the index level reflects only fair_value changes — not population growth or constituent additions/removals.

4.1 Inception

When an index first comes online (first date with ≥5 eligible constituents):

```
divisor(inception) = aggregate_effective_market_cap(inception) / 100
index_value(inception) = 100
```

All indices start at 100 on their inception date. This makes percentage changes immediately readable ("index is at 138" = up 38% since inception).

4.2 Daily Update — Pop Change Adjustment

When daily data arrives, before computing today's index value, check whether any constituent's population changed since yesterday. If yes, adjust the divisor so the population change does not artificially move the index level.

```
yesterday_baseline = sum over today's constituents of:
  today's fair_value × YESTERDAY's population × confidence_factor

today_actual = sum over today's constituents of:
  today's fair_value × today's population × confidence_factor

if today_actual ≠ yesterday_baseline:
  divisor(t) = divisor(t-1) × (today_actual / yesterday_baseline)
else:
  divisor(t) = divisor(t-1)
```

> The result: index value moves **only when fair_values change**. Population growth is absorbed into the divisor and reflected only in the redistribution of constituent weights for future moves.

4.3 Monthly Rebalance — Constituent Universe

On the last calendar day of each month, the constituent universe is re-evaluated:

- Re-check eligibility for every atom in the index's potential universe (matches scope + grade filter)
- Add newly-eligible atoms; remove atoms that have been ineligible for ≥1 month
- Adjust divisor so that the index value is preserved across the rebalance:
  `new_divisor = new_aggregate_market_cap / current_index_value × 100`
- Persist the rebalance event with full diff (atoms added, atoms removed) for audit

WEIGHTING

5. Confidence-Dampened Market Cap Weighting

Constituents are weighted by their effective market cap — a confidence-dampened version of raw market cap. Low-confidence atoms contribute less to the index without being fully excluded.

```
effective_market_cap = market_cap × √(confidence_score / 100)
```

The square-root function gives a gentle penalty: high-confidence atoms pass through nearly unchanged, while low-confidence atoms are meaningfully dampened.

Confidence Score	Weight Factor	Effective Contribution
100 (perfect)	1.000	100% of market cap
90 (very high)	0.949	94.9% of market cap
70 (high)	0.837	83.7% of market cap
50 (medium)	0.707	70.7% of market cap
30 (low)	0.548	54.8% of market cap
10 (very low)	0.316	31.6% of market cap

Rationale: a pure market-cap weighting treats a $100M atom with confidence 30 the same as a $100M atom with confidence 90. The first is much noisier and shouldn't carry the same weight in a published index. Square-root dampening lets the engine include all eligible atoms while making the noisy ones contribute proportionally less.

EXAMPLE

6. Worked Example — Two Days In Pokémon Base Set

Trace through 'Pokémon Base Set · 1st Edition Shadowless · PSA 9' with three constituents for clarity (real index has ~80-100).

Day 1 (Inception)

Constituent	Fair Value	Population	Confidence	Effective MCap
Charizard #4	$10,000	700	85	$6,460,300
Blastoise #2	$5,000	650	78	$2,869,300
Venusaur #15	$4,000	600	72	$2,036,800

Aggregate effective market cap = $11,366,400
Divisor = 11,366,400 / 100 = **113,664**
Index value = 11,366,400 / 113,664 = **100.00**

Day 2 — Prices Move, Population Unchanged

Constituent	Fair Value	Population	Confidence	Effective MCap
Charizard #4	$10,500	700	85	$6,783,315
Blastoise #2	$4,900	650	78	$2,811,914
Venusaur #15	$4,000	600	72	$2,036,800

Population unchanged → no divisor adjustment.
Aggregate = $11,632,029
Index value = 11,632,029 / 113,664 = **102.34**

Index up 2.34% — pure price movement, properly attributed.

Day 3 — Prices Unchanged, Population Grows

Constituent	Fair Value	Population	Confidence	Effective MCap
Charizard #4	$10,500	750 (+50)	85	$7,267,838
Blastoise #2	$4,900	650	78	$2,811,914
Venusaur #15	$4,000	600	72	$2,036,800

Population changed → adjust divisor first.

Yesterday baseline (today's FV × yesterday's pop):
Charizard: $10,500 × 700 × 0.922 = $6,783,300
Blastoise: $4,900 × 650 × 0.883 = $2,811,914
Venusaur: $4,000 × 600 × 0.849 = $2,036,800
Total: $11,632,014

Today actual: $12,116,552

Adjustment ratio: 12,116,552 / 11,632,014 = 1.04167
New divisor: 113,664 × 1.04167 = **118,400**
Index value: 12,116,552 / 118,400 = **102.34**

> Index level unchanged from Day 2 — the population growth was absorbed into the divisor. Charizard's weight in the index increased (from 58.3% to 59.9%), but the index level only reflects price action. **This is the discipline.**

SCHEMA

7. Database Schema

Four tables capture the full state of every index:

7.1 indices (definitions)

Column	Type	Purpose
id	BigSerial PK	Unique index identifier
slug	Varchar	URL-friendly identifier (e.g., 'pokemon-base-set-1st-edition-shadowless-psa-9')
name	Varchar	Display name
scope_type	Varchar	'set' or 'print_run'
scope_id	BigInt	set_id or printing_id (depending on scope_type)
grade_filter	Varchar	'composite', 'psa_10', 'psa_9', 'psa_8', 'psa_7'
set_id	BigInt FK	Always populated for hierarchy queries
inception_date	Date nullable	First date with ≥5 eligible constituents
current_value, current_divisor	Numeric(18,4)	Latest published state
constituent_count	Integer	Latest constituent count
last_rebalanced_at	Date	Most recent monthly rebalance

7.2 index_constituents (current membership)

Column	Type	Purpose
index_id, printing_id, grader_id, grade_id	FKs	Composite key (unique together)
effective_since	Date	When this atom joined the index
ineligible_since	Date nullable	First date of ineligibility (NULL if currently eligible)
last_weight	Numeric	Latest weight (for dashboard)

7.3 index_values (daily time series)

Column	Type	Purpose
id, index_id, as_of_date	BigSerial, FK, Date	Composite key
value	Numeric(18,4)	Index value at end of day
divisor	Numeric(20,4)	Divisor used for this day

Column	Type	Purpose
aggregate_effective_market_cap	Numeric(24,2)	For audit
constituent_count	Integer	How many atoms contributed
divisor_adjusted_today	Boolean	Was divisor adjusted on this day (pop change or rebalance)

7.4 index_constituent_contributions (audit trail)

Per-(index, constituent, date) row capturing each constituent's contribution. Stored for full audit; can be pruned to recent history if storage becomes a concern.

Column	Type	Purpose
index_id, printing_id, grader_id, grade_id, as_of_date	Composite PK	One row per atom per index per date
fair_value, population, confidence_score	Cached	Inputs as of this date
effective_market_cap	Numeric	fair_value × population × √(conf/100)
weight	Numeric	Effective market cap / aggregate (for dashboard)

7.5 index_rebalance_events (history)

One row per monthly rebalance per index, capturing what changed:

Column	Type	Purpose
index_id, as_of_date	Composite PK	One row per index per rebalance
divisor_before, divisor_after	Numeric	Divisor change at this rebalance
constituents_added	BigInt[]	Array of printing/grade tuples added
constituents_removed	BigInt[]	Array of printing/grade tuples removed
aggregate_market_cap_before, after	Numeric	MCap change pre/post rebalance

EDGE CASES

8. Edge Cases

Condition	Behavior
Constituent missing fair_value on a specific date	Skip that constituent for that date; index continues with remaining constituents
Population data missing for a date	Use latest known population (consistent with market_cap engine)
Fewer than 5 eligible constituents on a date	Don't publish index value for that date
Index newly comes online (inception)	Set value = 100, divisor = aggregate / 100

Condition	Behavior
Constituent loses eligibility (FV→NULL or pop→0)	Mark ineligible_since = date; remove at next rebalance after 1 month
Constituent regains eligibility within 1 month	Reset ineligible_since to NULL; stays in index
Index has no constituents (worst case)	Skip publish; index remains paused until breadth returns

OPERATIONS

9. Daily Compute And Monthly Rebalance

9.1 Daily Job (runs after market_cap calc completes)

- For each published index: load current constituents, check for population changes vs yesterday, adjust divisor if needed
- Compute effective market cap with confidence dampening
- Write to `index_values` and `index_constituent_contributions`
- Estimated runtime: ~30 seconds per 25 indices on hosted Supabase

9.2 Monthly Rebalance Job (last calendar day)

- Re-discover print runs from current data (handles new sets / printings being added)
- Re-evaluate eligibility for every potential constituent
- Add newly-eligible, remove atoms ineligible for ≥1 month
- Adjust divisor to preserve index level across the rebalance
- Write rebalance event to `index_rebalance_events`

9.3 Backfill

Initial backfill computes index values for every date from the earliest available `fair_value` through today. Indices come online on their natural inception date (first date with ≥5 constituents) and continue from there. Dates before any specific index's inception have no row for that index.

LIMITATIONS

10. Known Limitations

- **Historical population is single-snapshot.** The market_cap engine uses the latest population for all historical dates (~5-15% upward bias on older figures). Indices inherit this. Acceptable trade-off for now.
- **Raw indices deferred.** Raw card price methodology is unresolved (no clean 'raw fair value' equivalent and no population analog). Raw indices will be a separate ticket once raw_prices ingestion stabilizes and we design the analog math.
- **Auto-recompute on fair_value refresh is deferred.** If fair_value is recalculated for past dates (as happens during eligibility-set updates), the affected index dates need recompute. For now this is manual; automation comes in a separate operational ticket.
- **Half-grades (PSA 8.5, 9.5) excluded from composite.** Engine reads from grades table which holds integer values only. Half-grade sales contribute to fair_value at the parser level but don't form their own atom or affect composite weighting.
- **No concentration caps in v1.** A single atom (e.g., Charizard 1st Ed Shadowless PSA 10) might dominate its index. Cap-variant indices (with -D suffix) can be added in a future ticket if needed.
- **PSA only.** CGC and BGS atoms exist in market_cap but are excluded from indices. Cross-grader indices are a future enhancement.

This methodology document is the spec for ticket SCRUM-63 in the TCGBerg backlog. Implementation details (table DDL, module structure, test coverage) are captured in the ticket itself.

Companion documents: Fair Value Methodology (SCRUM-61), Market Cap Methodology (SCRUM-62).
Document version 1.0 — May 2026.