# Design Cube in Kylin

dev@kylin.incubator.apache.org

# Before You Start

- Kylin is a MOLAP engine on Hadoop.
- Understand Kylin helps cube design a lot.
  - http://www.slideshare.net/YangLi43/apache-kylin-deep-dive-2014-dec
- This deck summarizes best practices and patterns on how to design an efficient cube.
  - For detailed steps to create a cube, check out https://github.com/KylinOLAP/Kylin/wiki/Kylin-Cube-Creation-Tutorial

# Overview

- Identify Star Schema
- Design Cube
  - Dimensions
  - Measures
  - Incremental Build
  - Advanced Options
- Build and Verify

# Identify Star Schema

- Kylin creates cube from a star schema of Hive tables.
- One fact table that has ever growing records, like transactions.
- A few dimension tables that are relatively static, like users and products.
- Hive tables must be synced into Kylin first.

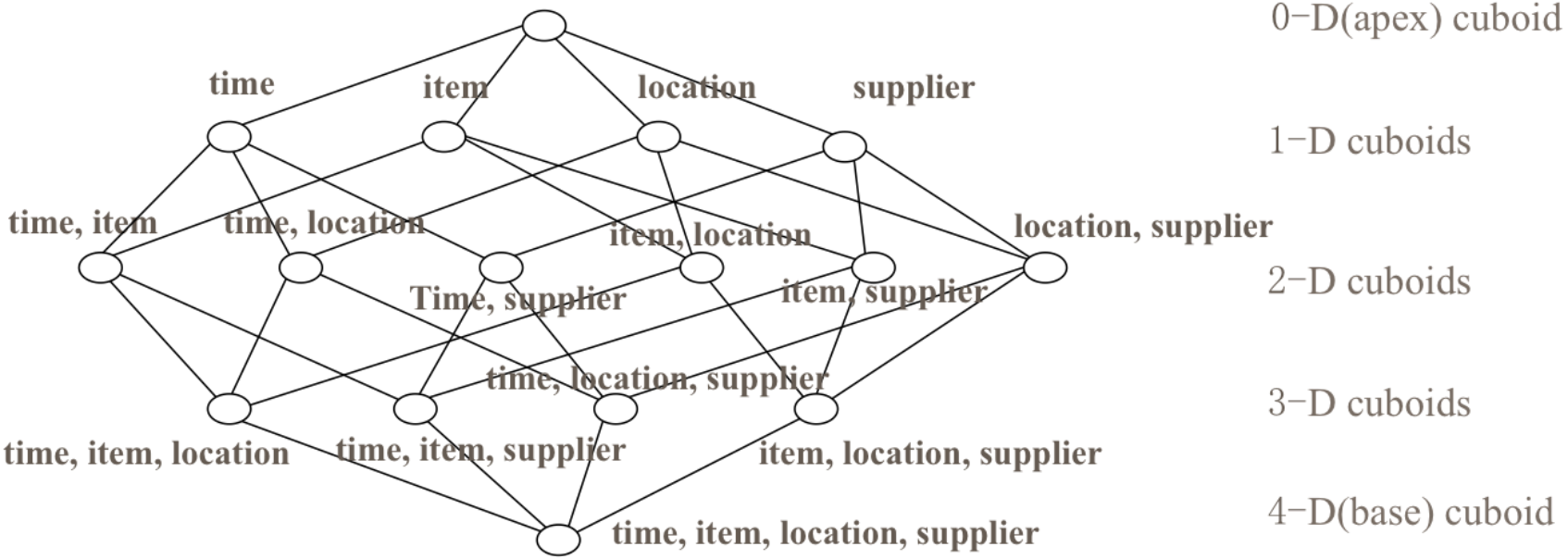
# Know Cardinalities of Columns

- Cardinalities have significant impact on cube size and query latency.
  - High Cardinality: > 1,000
  - Ultra High Cardinality: > 1,000,000

- Avoid UHC as much as possible.
  - If it's used as indicator, then put the indicator in cube.
  - Try categorize values or derive features from the UHC rather than putting the original value in cube.

# Cube Concepts

Cube = all combination of dimensions
Cuboid = one combination of dimensions
Curse of dimensionality: N dimension cube has $2^N$ cuboid



# Design Dimensions

- 15 dimensions or less is most ideal.
  - More than that causes slowness in cube build and query latency.
  - Does user really need a report of 15+ dimensions?
  - You can define multiple cubes on one star schema for fulfill different analysis scenarios.

- Control the total number of dimensions.
  - Mandatory dimension
  - Hierarchy dimension
  - Derived dimension

# Mandatory Dimension

- Dimension that presents in every query.
  - like Date

- Mandatory dimension cuts cuboid combinations by half.

Normal Dimensions

| | | |
|---|---|---|
| A | B | C |
| A | B | - |
| - | B | C |
| A | - | C |
| A | - | - |
| - | B | - |
| - | - | C |
| - | - | - |

A is Mandatory

| | | |
|---|---|---|
| A | B | C |
| A | B | - |
| A | - | C |
| A | - | - |

# Hierarchy Dimension

- Dimensions that form a “contains” relationship where parent level is required for child level to make sense.
  - like Year -> Month -> Day;  or  Country -> City

- Hierarchy dimension reduces combination from $2^N$ to N+1.

Normal Dimensions

| | | |
|---|---|---|
| A | B | C |
| A | B | - |
| - | B | C |
| A | - | C |
| A | - | - |
| - | B | - |
| - | - | C |
| - | - | - |

A->B->C is Hierarchy

| | | |
|---|---|---|
| A | B | C |
| A | B | - |
| A | - | - |
| - | - | - |

# Derived Dimension

- Dimensions on lookup table that can be derived by PK.
  - like User ID derives [Name, Age, Gender]

- Derived dimension reduces combination from $2^N$ to 2 at the cost of extra runtime aggregation.

Normal Dimensions

| A | B | C |
|---|---|---|
| A | B | - |
| - | B | C |
| A | - | C |
| A | - | - |
| - | B | - |
| - | - | C |
| - | - | - |

A, B, C are Derived by ID

| ID |
|---|
| - |

# The Order of Dimensions

- Finally, define dimensions in following order.
  - Mandatory dimension
  - Dimensions that heavily involved in filters
  - High cardinality dimensions
  - Low cardinality dimensions
- Filter first, helps to cut down of query scan ranges.
- High cardinality first, helps to calculate cube efficiently.

# Define Measures

- Kylin currently support
  - Sum
  - Count
  - Max
  - Min
  - Average
  - Distinct Count (based on HyperLogLog)

- Distinct Count is a very heavy data type.
  - Error rate<1.22% takes 64KB per cell.
  - Convince user to use the wildest tolerable error rate.
  - Distinct Count is slower to build and query comparing to other measures.

# Incremental Build

- Kylin supports incremental build along a time dimension if enabled.
- Setting a start time, cube segments can be built daily (or any period) processing only the incremental data.
- A segment can be refreshed relatively cheaply to reflect changes in hive table.
- With the increasing number of segments, query would slow down a bit.
- Merge segments to control the total number < 10 for best performance.

# Advanced Options

- Leave advanced options as is if you are not sure what they mean.

- Aggregation groups give finest control on which cuboids to build.
  - Partial cube -- Only combinations within the same group are built.
  - For cube with 30 dimensions, if divide the dimensions into 3 groups, the cuboid number will reduce from 1 Billion to 3 Thousands.
    - $2^{30}$ => $2^{10} + 2^{10} + 2^{10}$
  - It's tradeoff between online aggregation and offline pre-aggregation.

- Rowkeys, suggest leave them untouched.

# Build and Verify

- Once the cube is created, build it, and ready to verify.

- Notes on the SQLs
  - Write queries against the star schema, cubes are transparent at the query time.
  - Sanity check: `select count(*) from fact`
  - Make sure the join relationships (inner or left) matches the cube definition exactly.
  - Kylin works best with a `group by` clause.
  - Date constant is like `date '1970-01-01'`

# Q & A

Thanks!