**Types of fields in a Dimension**

■ Keys - Used to identify entities

■ Name columns - Used for human names of entities

■ Attributes - Used for pivoting in analyses. Continuous values should be discretized according to business rules.

■ Member properties - Used for labels in a report such as company name, address etc. Not pivoted on

■ Lineage columns - Used for auditing, and never exposed to end users

**Hierarchies**

Dimension can have natural hierarchies that provide drill-down path. Hierarchies have levels which have members eg Jan, Feb, March etc.

In a snowflake schema, hierarchies are implemented as joined look-up tables.

In a star schema, they are implemented in terms of attributes

**Slowly Changing Dimensions**

Type 1 SCD – update the single row. Used when there is no concept of history for that attribute eg product name is changed there is no sense analysing the old name

Type 1 SCD – create a new row. Used when there is a concept of history eg someone moves from one address to another. If doing this then need a surrogate key because there will be duplicates.

**Types of fields in a Fact table**

Measure – normally numerical, can be aggregated and what is being measured eg sales value.

Foreign keys to dimensions. – all foreign keys together usually uniquely identify a row.

Linage

**Additivity of Measures**

Additive – measure can be summed across any dimension eg sales amount

Non-Additive – can’t be summed across any dimentions. Normally averaged eg unit price

Semi-Additive – can be summed across any dimension except time. Eg balance

**Related fact tables**

Fact tables may be related and so one may be on ‘many’ side. This won’t work for SSAS so can put in an intermediary Dimension table. Not derived from source, is a construct of the data warehouse.

**Calculating space needed by DW**

For data files, extrapolate from a small period and add 25% for rebuilding indexes without fragmentation.

For log, find your biggest transaction

**Files Groups –** have one group per partition. Read this when have a chance

https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee796978(v=cs.20)

**DW schemas** – one for ETL one for the DW

**Sequence** – are independently created from a table and can be used by multiple tables. Can perform better and offer more functionality

**Computed columns** – can be used to discretized columns

**Fact tables – foreign keys** – can be disabled for loading into but ETL should be robust enough to error if violations occur. Or it can add a row when it doesn’t exist. This is called **inferred member**.

**Indexes**

Clustered Index, why?

Can defrag the table

Move table to a new file groups without having to drop and recreate

By default, a non-clustered index will be created which will probably be bigger than a clustered index.

Can create covered queries that run only off the index.

Opt for an integer autonumbering surrogate key as the clustered primary key for all DW tables, unless there is a really strong reason to decide otherwise.

Non-clustered index, won’t be used much on dimension attributes. Could be used on name columns and member property columns, used in parametrised reports eg company name.

Probably don’t need indexes on the foreign keys, analyse type of join being done eg is it hash or merge?

**Indexed Views**

Optimize queries that aggregate data and perform multiple joins. They basically material a view and will be used if a query

**Compression** – useful when data is mostly read and rarely updated

Row compression…

*Row compression*reduces metadata overhead by storing fixed data type columns in avariable-lengthformat. This includes strings and numeric data. Row compression has only a small impact on CPU resources and is often appropriate for OLTP applications as well.

Page compression…

*Page compression*includes row compression, but also adds prefix and dictionary compressions.*Prefix compression*stores repeated prefixes of values from a single column in a special compression information (CI) structure that immediately follows the page header, replacing the repeated prefix values with a reference to the corresponding prefix.*Dictionary compression*stores repeated values anywhere in a page in the CI area. Dictionary compression is not restricted to a single column.

Unicode compression – stored Unicode strings as single bytes where able to

**Columnstore index**

Stores indexes by column rather than row. Not good for very selective queries.  Not suitable for OLTP workloads. And force table to be read-only. Must drop and recreate. One columnstore index per table. So

Use columnstore indexes for

■■Read-mostlyworkloads.

■■Updates that append new data.

■■Workflows that permit partitioning or index drop/rebuild.

■■Queries that often scan and aggregate lots of data.

Don’t use columnstore indexes when

■■You update the data frequently.

■■Partition switching or rebuilding indexes doesn’t fit your workflow.

■■Your workload includes mostly small lookup queries.

**Batch mode processing**

Retrieves data in batches rather than individual rows. Lowers CPU. Useful for bitmap filtered hash join and scan operators

**Partitioning**

Tables and indexes can be partitioned. Often done on dates.

Partition function - An object that maps rows to partitions by using values from specific columns (partitioned columns)

Partition scheme – Maps partitions to filegroups

Aligned index – indexes that have the same portioning as base table. Columnstore indexes **have** to be aligned with their base tables

Partition elimination – query optimizer only access partitions it needs to

Partition switching - This is a process that switches a block of data from one table or partition to another table or partition.

If you want to switch content from a nonpartitioned table to a partition of a partitioned table, what conditions must the nonpartitioned table meet?

It must have the same constraints as the partitioned table.

It must have the same compression as the partitioned table.

It must have a check constraint on the partitioning column that guarantees that all of the data goes to exactly one partition of the partitioned table.

It must have the same indexes as the partitioned table.

**Data Linage**

Many in-built functions in SSIS and T-SQL for logging linage eg

APP\_NAME() - ApplicationName

DATABASE\_PRINCIPAL\_ID() DatabasePrincipalId

USER\_NAME() DatabasePrincipalName

SUSER\_SID() ServerPrincipalId

SUSER\_SNAME() ServerPrincipalName

CONNECTIONPROPERTY('net\_transport') transport protocol

CONNECTIONPROPERTY('client\_net\_address') Client net address

CURRENT\_TIMESTAMP

@@ROWCOUNT – number of rows process