**Nominal data**:由数字代表类别1 = white, 2 = blue, 3 = yellow

**Dichotomous Data**: 二分数据 eg. 1 = true, 0 = false;

**Ordinal data**: 数字代表非量化大小的数据1 = low, 3 = high

**Interval data**: 没有绝对零值的数据,温度(0度!=绝对0度)

**Ratio data**: 有绝对零值的数据 eg. 长度

**Lower/upper inner fence**: Q1–1.5IQR/Q3+1.5IQR

**Box plot**:for distribution, **correlation**:for dependence,use parametric test for normally distributed, non-parametric for ratio and ordinal

**WEEK3: Accessing data in relational databases; intro to SQL**

**Database:**collection of logically related data and its description, have entities,attributes,relationships.

**WHY**:data size;data update;accuracy;security;redundancy

**File process disadvantage**: Program-Data Dependence, Data Redundancy-Integrity problem, Limited Data Sharing,long develop time,maintenance

**dependency problem**:each man maintain their own data, each app need metadata of each file&own processing routines, lack of coordination ctrl, non-standard format

r**edundancy problem**:waste space,maintenance,hurt integrity

**database advantage**:independ app&data-less develop time,ez maintenance,efficient;less redundancy;data sharing; standards; data quality; better responsiveness,securety&recovery

**DB concept**:table,field(attribute),field data type, Primary Key,record(tuple,row)

**WEEK4: Declarative data analysis**

**DDL**:data definition language, for create/alter relation schema

**DML**:data manilulation language, for retrieval data(query)

**PK**: unique, minimal identifier of a relation

**FK**: identifiers with dependent relation to refer parent relation

**Clean data step**:type&name conversion,filter missing & inconsistent data,unify semantic data,match different source entries,rescaling&dimensionality reduction

**ETL**:capture/extract-data cleansing-transform-load

**Fact table:data modeling**, numeric measures depend on set of dimensions.

**DB is for**: shared, persistent storage of structured data, consistent updating

**WEEK5: Indexes and Data Partitioning**

**Storage Hierarchy**: **primary**:fast,volatile,used for current data; **secondary**:non-volatile,on-line,for main DB; **tertiary**:slowest,off-line,used for archive

**Access**:seek time 1-20ms,rotation delay 0-10ms,transfer time 1ms/4kb,try reduce seek&rotation time

**Buffer**:store missing page between memory&disk transfer

**File organization**:**heap file**: unordered place in wherever have space, need linear scan-read all files, leave gaps when delete; **Sorted file**: rows sorted based on attribute, successive rows on successive pages; use binary search to access, hard to maintain

**Index**: search key-index entries-data records-matching record

**Ordered**:search keys in sorted order;

**hash**:keys distribute uniformly with hash function

**downside**:additional IO, need update, user decision

**cluster index**:index entries and rows order same way, at most 1 in a table, CREATE TABLE create this with primary key, good for range search-use index to get first record in range

**uncluster index**:index entries & rows not order same way, can be many secondary indices on a table, CREATE INDEX,for keys other than primary key

**cover index**: index that contains all attributes required to answer a given SQL query Prefix of the search key must be the attributes from the WHERE

**data partition**:hori:横切, vert:纵切, good for back up&manage

**WEEK6:TEXT DATA PROCESSING**

**text data:** unstructured, text heavy, end up ambiguities

**machine learning:** **supervised**:prediction, classify; **unsupervised**:cluster, find association; **semi-supervised**; **reinforcement**: decision making

**spam detection**: supervised, text-feature vectors-machine learning algorithm+labels –model

**feature vectors**: Represent document as a multiset of words, Keep frequency information, Disregard grammar and word order

**tokenization**: split string into tokens. Eg: “Friends, Romans, Romans, countrymen” - [“Friends”, “Romans”, “Romans”, “countrymen”]

**normalization**: Map similar words to the same token, include Stemming&lowercasing. Eg [“Friends”, “Romans”, “Romans”, “countrymen”] - [“friend”, “roman”, “roman”, “countrymen”]

**term frequency(TF)weighting**: higher frequency = more weight.. Eg: [“friend”, “roman”, “roman”, “countrymen”] - {“friend”: 1, “roman”: 2, “countryman”: 1}

**inverse document frequency(IDF)**: Give less weight to terms that are common across documents, IDF = log(|total docs|/|docs containing term|)

**TFIDF weighting**: TFIDF = TF \* IDF. Eg: [“friend”, “roman”, “countrymen”] - {“friend”: 0.1, “roman”: 0.8, “countrymen”: 0.2}

**Dimensionality**: more feature -> worse performance, training required grow exponentially.

**Text driven forecast**: Given text T link to a phenomenon, make prediction about a measurement M of that phenomenon, obtainable only in future.

**Natual language process**: understanding(tokenization, POS tagging, parsing) – generation – summarization.

**Information extraction**: entity recognize&disambiguation, relation extraction.

**knowledge base population**:-entity linking&slot filling.

**WEEK8:TIME SERIES DATA**

**Temporal data types: instant/poin**t: DATE(day, month, year), TIMESTAMP(date +time), TIME(no date only time); interval(a duration), period(anchored duration)

**Kinds of time**: **User-defined time**: an uninterpreted time interval; **Valid Time**: the time when a fact is true(Can move forward and backward); **Transaction time**: history of database activity, only move forward

**Represent ‘till now’**: use max-timestamp

**Period**: added as a half-closed interval [Estart, EEnd)

**Timeseries data**: **Point-based**: Multiple rows with atomic data types, need multi tuple for long lasting fact; **Sequence-based**: Single row with array of time point, Requires array\_agg()&UNNEST (例子见最后)

**Wk 9: Web Scraping Data and Semistructured Data**

**Web scraping steps: Reconnaissance**: Identify source, check its structure and content; **Webpage retrieval**: Download pages from source, in a script auto-generates new URLs based on website structure and its URL format; **Data extraction**: Content parsing, raw data extraction; **Data cleaning**: transformation into required format; **Data storage**

**Hint:** note URL format, note webpage structure.

**Data cleaning:** Cross-check data consistency once loaded

**Webservice api:** many APIs require authentication

**Semistructured data:** irregular attribute, nesting structure objects, diff types in diff obj, heterogeneous collections.eg: XML, json.

**XML:** describes content, specify general syntactic structure, elements, attributes, character set, DTD. **well-formed**: document satisfies XML syntax constraints (matching tags); **valid**: document is well-formed and satisfies a pre-defined schema, as formalised in a document type definition (DTD); stronger in the database area

**JSON:** nested key-value pairs with types Number, String, Boolean & Array, low-overhead.

**Semi structure vs structured data:** **Relational**: Schema-first, rich type system for attributes, integrity constraints; only atomic type attributes allowed-> many different tables, joins needed; **Semi-structured**: Self-describing data with flexible structure; Nested data model with tree-structure; optional attributes, grammar, schema and vocabulary

**Storing semi-str**:**XML**: XML data type&support functions, Not stored as a string, but as tree structure. Supports navigation via storage and indexing; **JSON:** JSON type stores exact copy of JSON data in attribute, JSONB type stores a binary, decomposed versions, More overhead for inserts, but claims significant faster querying

**WEEK10:geo-spatial data**

**Spatial data**: entities have a location, identify their location.

**SDBMS vs GIS**: **SDBMS**: Handle large spatial data in secondary storage, semantics built into query language, Specialized index structure for spatial data; **GIS**: SDBMS Client, Characterized by geographic analysis functions, organized in “layers”

**Point data:** Points in a multidimensional space, eg: trajectory data, raster data.**Region data:** Objects have spatial extent with location and boundary, geometric approximations constructed using line segments, polygons, etc., called vector data.

**Field-based model:** Spatial Framework is a partitioning of space, Field Functions: f: Spatial Framework -> Attribute Domain, Field Operations like addition(+) and composition(o)

**Object model:** **Objects**: distinct identifiable things relevant to an application,have attributes and operations; **Attribute**: a simple property of an object; **Operation**s: function maps object attributes to other objects

**Classify**: **simple**: 0d-point, 1d-curve, 2d-surface ;; **collection**: Polygon collection e.g. boundary

**Coordinate sys**: Each instance of a spatial type has a SRID, SRID is an integer that identifies the coordinate system in which the type is described; List typically according to prevalent standards (EPSG/OGP);**types**: Cartesian Coordinates: point positions from a defined origin along axes on a plane, Geodetic Coordinates: angular coordinates (longitude and latitude), Projected Coordinates: planar Cartesian coordinates from Earth surface to plane

**Topology**: invariant under elastic deformation, have interior, boundary, and exterior for an object A.

**9intersection model**: relationships of A and B arranged into 3\*3 matrix, with value 0&1

**Spatial queries**: Spatial Range Queries, Nearest-Neighbor Queries, Spatial Join Queries

**Processing**: Spatial Join(polygon<->point); ‘filterrefine’ strategy: Objects with minimum bounding box (MBR) intersecting query regions, Query region really intersecting only with B and C;indexing

**Rtree**: tree-structured index remains balanced on inserts and deletes; Each key stored in a leaf entry is intuitively a box; Leaf entry = < n-dimensional box; Non-leaf entry = < n-dim box; All leaves at same distance from root; Nodes can be kept 50% full

**Spatial api**: GeoJSON, KML; to ingest, look up, or visualize data

**WEEK11 IMAGE PROCESSING**

**Type : RGB**: Each pixel has a particular color, described by amount of pixels in 3 channels. **Gray-scale**: 1 channel, Each pixel is a shade of gray. **Binary image**: Each pixel is black (0) or white (1), 1 channel, ‘mask’ in the image processing domain

**Process**: enhancement, restoration, segmentation (subdividing an image into constituent parts.

**Steps**: remove noise, Extraction of region of interest (ROI) using binary mask- intensity->no enhance, Measurements

**Morphological**: probe an image with a small template called a structuring element-a small matrix of pixels, each with a value of zero or one;dilation: value of the output pixel is the maximum value of all the pixels in the input pixel's neighbourhood; **Erosion**: value of the output pixel is the minimum value of all the pixels in the input pixel's neighborhood

**Connectivity**: 4(十字), d（对角）, 8（圈）

**Enhance**: denoising, PSF, deconvolution

**Thresholding:** creates binary images from grey-level, <T = 0, >T = 1**.issue**: different gray levels, Noise.local: diff T for each pixel.

**Histogram**: frequency each intensity value in image occurs

**WEEK12 BIG DATA**

**3V**: volume, variety, velocity, need scalable platform&process abstraction (scale up大 vs scale out多)

Scale-Agnostic management: sharding- performance, replication –availability

**MapReduce**: Scan large volume data-> **Map**: Extract information-> Shuffle and sort intermediate results-> **Reduce**: aggregate intermediate results-> Generate final output; Key idea: provide an abstraction; **Pros**: flexible, gud scalability, ez parallelism, fault tolerance; **cons**: need programing skill, low level, batch-processing

**Iterative**: data mining and machine learning algorithms rely on global state and iterations

**Spark**: main-memory caching, more high-level data flow control; **flink**: build-in dataflow optimiser and pipelined processing

**SQL**

**Order**:SELECT \* FROM T ORDER BY a:add DESC for

**only show n entries**: SELECT FROM LIMIT n

**Compare**: BETWEEN a AND b,

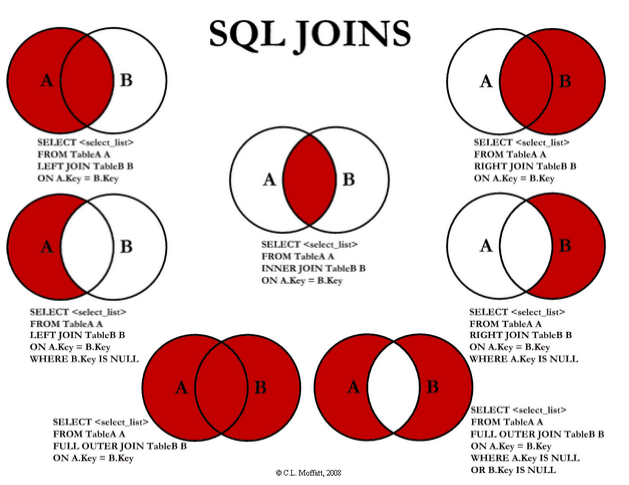
**NULL**:IS null/ IS NOT null

**Mode number**:MODE() WITHIN GROUP (ORDER BY attr)

**Median**:PERCENTILE\_DISC(0.5) WITHIN GROUP (ORDER BY attr)

**Join**:SELECT \* FROM a INNER JOIN b ON a.xx = b.yy

**add value based on c** :SELECT a FROM b GROUP BY c

**natual join**: SELECT \* FROM a NATURAL JOIN b, combine rows value agree in all common attributes 

**load csv**: COPY tablename FROM filename CSV [HEADER] [NULL '…']

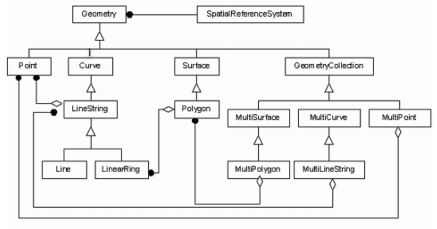
**index**:CREATE INDEX StudentName ON Student (name);;DROP INDEX indexname

**datarange table:** xxx DATERANGE, INSERT INTO Emp VALUES ('[2010, 2014]');, SELECT '[2010, 2013)' + '[2012, 2015)'

**index on timerange**: CREATE INDEX Emp\_idx ON Emp USING GIST (EPeriod)

**array:insert**: INSERT INTO xxx VALUES(1,'{2,3,4}'); **update:** UPDATE xxx SET num[4] = 2 WHERE id=..; **unnest:** SELECT UNNEST(temp) FROM Obs2 WHERE ..

**mapreduce:** SELECT out\_key, reduce(out\_value) FROM map(InputData) GROUP BY out\_key

**openGIS data model:** 

**9交图**

