**- DBMS Uses**

* Entity/ relationship/ weak entity
  + Weak entity: defined dependently on other entity (primary key contains foreign key)
  + Both Entity sets and Relationships become relations(tables)
* Draw ER diagram
  + Circle – attributes (underline for primary key)
  + Rectangle – entities
  + Diamond – relationship (need to mark “Many-to-many” or “many-to-one”)
* Physical/logical independency
  + The [logical](https://en.wikipedia.org/wiki/Logical) structure of the data is known as the 'schema definition'. In general, if a user application operates on a subset of the [attributes](https://en.wikipedia.org/wiki/Attribute_(computing)) of a [relation](https://en.wikipedia.org/wiki/Relation_(database)), it should not be affected later when new attributes are added to the same relation. Logical data independence indicates that the conceptual schema can be changed without affecting the existing schemas.
* Functional dependency

**- SQL**

* Table constrains:
  + Primary Key
  + Foreign key (be like “FOREIGN KEY (grader) REFERENCES Students (sid))”)
    - On delete/update restrict: child table prevents the update or delete from the parent table
    - On delete/update cascade: child table makes corresponding changes
  + Not NULL
  + Unique
  + Check

**- Tree-based Storage Structures (B+ Tree)**

# of Comparisons = ~log\_2(N) (depends on fan out)

**- Query execution algorithms**

Aggregations should be in the same step as “Group by”

**- Query Optimization**

External Merge/Sort

Joins

* Nested loop join
* Index nested loop join
* Merge join
* Hash join

Indexing (indexes are in memory or in disk?)

* B+ Tree index – better for sequential search
  + Good for range search, sequential search
* Hash index
  + good for equality search (random search)
  + not so good for range search (use tree indexes instead)
* Data Entry Representations:
  + Store records
  + Store < key values, Rid>
* Clustered index
  + If order of data records is the same as, or `close to’,

order of index data entries, then called clustered index

* We can have at most one clustered index on a data file.
* Unclustered index

A close up of a map

Description automatically generated

* Primary Index:
  + index key includes the file’s primary key
* Secondary Index: other indexes

- **Transaction Management**

* ACID
  + Atomicity: “ all or nothing”
  + Consistency: “it looks correct to me”
    - If each txn is consistent and the DB starts consistent, then it ends up consistent.
  + Isolation: “as if alone”
    - Execution of one txnis isolated from that of other txns.
  + Durability: “survive failures”
    - If a txncommits, its effects persist.

Recovery

* Write-ahead log
  + DBMS must write to disk the log file records that correspond to changes made to a database object before it can flush that object to disk.
* Steal policy:

Whether the DBMS allows an uncommitted txn to overwrite the most recent committed value of an object in non-volatile storage.

* + STEAL: Is allowed.
  + NO-STEAL: Is not allowed.
* Force policy:  
  Whether the DBMS requires that all updates made by a txn are reflected on non-volatile storage before the txn is allowed to commit.
  + FORCE: Is required.
  + NO-FORCE: Is not required.
* Traditional recovery:
  1. Log analysis: determines what pages are out-of-date
  2. Redo phase: redo all txn after check point
  3. Undo phase: roll back loser txn (txn that were not committed before crash)

**- Concurrency Control**

* Exclusive lock
* Shared lock

**- Columnar Storage**

* Row storage
  + better when work on the whole row
* Column storage
  + better when less columns and many rows are needed
  + better for updating columns, aggregations, selecting a few columns, and any other operation, stored procedures, etc. which work on the columns only.
  + Columnar Databases are quite slow for insertions as you insert a whole row

More: <https://medium.com/@mangatmodi/rowise-vs-columnar-database-theory-and-in-practice-53f54c8f6505>

**- Scaling & parallel query execution**

* Pipelining: Horizontally split query execution to different machines
* Partitioning: veristically split data to different machine and every machine conduct the same query execution plan.
* Log shipping: Sharing log files with different machines for transaction back-up
* Mirroring: make multiple physical copies
* Distributed Database
  + 2-phase commit
    - Each node records the outcome of each phase in a non-volatile storage log. What happens if coordinator crashes? → Participants must decide what to do. What happens if participant crashes? → Coordinator assumes that it responded with an abort if it hasn't sent an acknowledgement yet.