1 Disorder is a friend of scaling

1.1 Streaming through RAM

- Instead of reading and writing a single item at a time, batch things up
- When input buffer consumed, read another chunk. When out buffer fills, write to output using f(x) in the middle
- We can simply partition the input and parallelize

1.2 Rendezvous

- Streaming one chunk at a time is easy, but some algorithms need certain items to be co-resident in memory (not guaranteed to appear in the same input chunk)
- **Time-space rendezvous**: in the same place (RAM) at the same time - most of computing is about this
- Divide and Conquer: you have B chunks, use 1 for read into and one for write into, leaving B-2 chinks of RAM left as space for rendezvous

2 SQL

The standard for database queries - has been around since the 70s and remains the top language

2.1 Pros and Cons

Pros: SQL is a declarative language and is widely implemented (varying levels of efficiency and completeness). It is also general-purpose and feature-rich (many years of added features, extensible: callouts to other languages, data sources)
Cons: Constained (Core SQL is not a Turing-complete language - extensions make it Turing complete)

2.2 Relation Terminology

- Database: set of relations
- Relation (Table): Schema (description, schema of database is set of schema of its relations), Instance (data satisfying the schema)
- Attribute (Column)
- Tuple (Record, Row)

- Schema is fixed, set up in the beginning. Attribute names, atomic types. Populated with data that changes over time
- Instance can change: a multi-set of rows. Tables are unordered and there can be repeats
- DDL: Data Definition Langue is where you declare hat you want your tables to be and look like.
- DML: Data Manipulation language is where you say what info you actually want in the table
- RDBMS is responsible for efficient evaluation

Foreign Key (sid) References Sailors: this means that the sid will be tied to the sid in the Sailors table - you can't put something in the new table that didn't exist in the other table

2.3 Single Relation Queries

DISTINCT means remove duplicates SELECT [DISTINCT] what you want
FROM table
WHERE condition
GROUP BY column list
HAVING predicate
ORDER BY column list

Aggregates compute a summary of some arithmetic expression and produce one row of output
Group by partitions the table into seperate areas and produce aggregate result per group - cardinality of output is the number of distinct group values
Having predicate is applied after grouping and aggregation - hence can contain anything that could go in the SELECT list - only used in aggregate queries

Order: FROM comes first - gives us the table. Then we look at the table data and stream it through the WHERE clause. In the SELECT clause we see which columns we actually care about. Then we do GROUP BY and create buckets and collect tuples for each bucket. Then we look at the HAVING clause and throw out any bucket that doesn't satisfy. Then we eliminate duplicates if DISTINCT is on and then we do that AGGREGATE

2.4 Querying Multiple Relations - Join Queries

FROM Sailors S, Reserves R - we have multiple tables in this clause
WHERE S.sid = R.sid - otherwise we get nonsense

Range Variabels: needed when ambiguity could arise - same table used multiple times in FROM

Where S.name LIKE 'P(underscor)p(percent sign)' means name starts with a P then some stuff then a lower case p
EXCEPT can remove sailors who have reserved a boat - removing some group - also eliminates duplicates

2.5 Nested Queries

WHERE S.sid IN
For each S.sid check if it is in the nested table (can also do NOT IN)

WHERE EXISTS (subtable) - subtable will include WHERE S.sid = R.sid

WHERE S.rating ¿ ANY - where the rating is bigger than some selected rating value in the subtable

2.6 NULL Values

Field values that are sometimes unknown or inapplicable - SQL provides a special value null for such situations
In a truth table - ? or True = True, ? and True = ?, etc. - these are unknown values

2.7 Inner Join and Outer Join

FROM Sailors S
INNER JOIN Reserves r
ON s.sid = r.sid
ON is called the join condition

FROM Sailors S
LEFT OUTER JOIN Reserves r
ON s.sid = r.sid
We return all the matching rows, plus all unmatched rows from the table on the left of the join clause - use nulls in fields of non-matching tuples
Full Outer Join will use nulls to fill in blanks from both tables

2.8 Views: Named Queries

CREATE VIEW viewName
AS selectStatement
Views make development simpler, used for security, not "materialized" (you just store the select statement - you don't store the actual table)

GRANT privileges ON object TO users
Object can be a Table or a view, Privileges talk about operations allowed on

those table - Select, insert, delete, reference (create foreign key that references specified columns), all - can later be revoked

2.9 Subqeiers in FROM

Akin to a view
FROM Boads2 b,
(SELECT b.bid, COUNT(*)
FROM ... WHERE ...) AS Reds(Bid, scount)
Now you can refer to Reds later - this is a view created on the fly

2.10 Constrains

Integrity Constrains: conditions that every legal instance of a relation must satisfy (similar to assertions in other languages)
Inserts/deletes/updates that violate IC's are disallowed - can ensure application semantics (sid is a key) or prevent inconsistensies (sname must be a string)

Domain Constrians: field values must be of the right type (always enforced)

Primary Key and Foreign Key Constraints: keys are way to associate tuples in different relations - one form of ICn. A foreign key is when a key references a table where it isn't the primary key (sid primary key in students table but foreign key in classes table)