Lec 1 Introduction

Logistics

- Tradition — Class DJ ~~
- Policies & schedule → Website
- Discussion & announcement → Piazza
- Homework & Projects → Gradescope
- Grades → Canvas
- Lang — C++ is prerequisit

Project 0 - C++
- CRDTs

DB Sys background

Collection of inter-related data, organised s.t. some operations are efficient

Ex.
- SQLite — used a lot, compiles to application code
- CSV — some structure... but maybe inefficient

```
"name1", 1992, "USA"
"name2", 1993, "UK"
```

Can write programme to parse the CSV, search through rows, ...

→ Better organisation needed!
Initial goal: digitise paper files

Issues
- Data integrity
  - Make sure right format
  - Make sure invariants respected
  - Data makes semantic sense
  - Correct type
- Efficiency
  - O(lg n) search ?
  - Multi-core support
- HCI
  - Interact with end-user
  - Multi-user access & concurrency
  - Abstractions to simplify access
- Durability
  - Anti file corruption
  - File size scaling

Relational Model

Data model — set of concepts for describing the data in DB
Schema — description of a particular data collection using some data model

Ex.

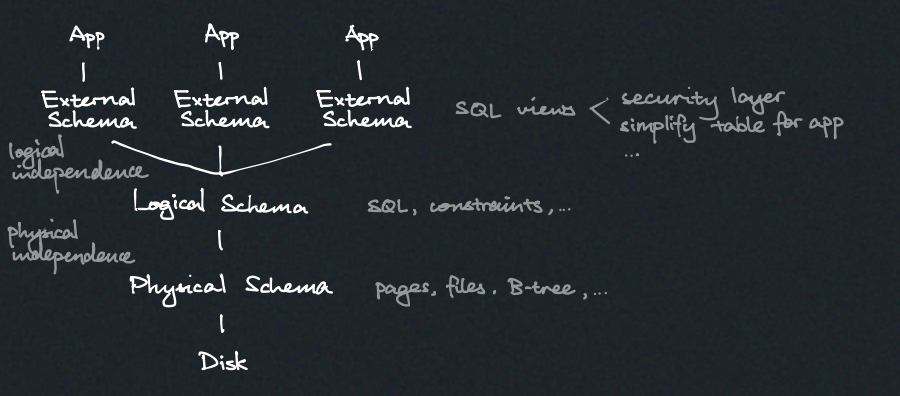
```
CREATE TABLE Artist(name VARCHAR, year DATE, country CHAR(60));
CREATE TABLE Album(Albumid INTEGER, name VARCHAR, year DATE);
```

Types of models
- Relational] our focus
- Key-val] NoSQL
- Graph
- Document / Object / XML
- Wide-column / Column-family
- Array / Vectors / Matrix] New, ML useful?
- Hierarchical] obsolete?
- Network
- Multi-value

History
- ... early ones needed lots of manual work
- Ted Codd, 1969, IBM Research, Relational Model
  - X Relate by pointer & chase down pointers
  - ✓ Mathematical model for relation
    - Set semantics
  - CODASYL

    Idea:
    - DB should be in relations data structure
    - Disk storage is impl defined
    - High-level interface abstracted away from users

Abstraction levels



Relation — unordered set of tables of tuples
Tuples (a row) — set of attribute values
NULL value ∅ — missing info
Primary key — unique id
- SQL does this automatically
- maintains uniqueness

Foreign key — reference some primary key in other table
Ex. ArtistAlbum(artist_id, album_id)

Constraints — user defined invariants
Ex. name VARCHAR NOT NULL
CHECK (date_trunc('year', year) = year)
check that year in the year field is proper year

Relational Algebra

Data Manipulation Languages (DML)
- Procedural (relational algebra)
- Declarative (calculus-based)

Algebra construct

σ	select	−	difference
π	projection	×	product
∪	union	⋈	join
∩	intersection		

SELECT $\sigma_{predicate}(R)$

$\sigma_{id="a2"}(R)$ $\sigma_{id="a2" \wedge b>6}(R)$

PROJECTION $\Pi_{A_1,\dots,A_n}(R)$
- rearrange attributes
- remove attributes
- manipulate values for derived attributes

$\Pi_{a\_id+3,\ b\_id}(\sigma_{a\_id>3}(R))$

UNION $R \cup S$
- combine relations with same schema
- keep / remove unique

PRODUCT $R \times S$
- takes cartesian product

```
SELECT * FROM R CROSS JOIN S;
```

JOIN $R \bowtie S$
- do cartesian product but only only keep those with ID matches

R(a_id, b_id)
S(a_id, b_id, val)

```
select * from R natural join S;
select * from R join S using (a_id, b_id);
select * from R join S on R.a_id = S.a_id
    and R.b_id = S.b_id;
```

Other ones...

ρ	rename
R←S	assignment
δ	remove duplicate
γ	aggregate
τ	sort
R÷S	divide

→ With these algebra, we can do query optimisation

Alternative Data Models

Document-based
- simple

Just store Json objects

```
Artist {
  int id;
  str name;
  int year;
  Album albums[];
}                    ← then populate this
Album {
  int id;
  str name;
  int year;
}
```

Vectors: text → Transformer → vector → vertex index

Conflict-free Replicated Data Type (CRDT)

Problems
- Distributed system, each device making local changes but the system's global state should converge

Ex. CRDT counter btwn 3 nodes 

→ Turn counter into monotonic state

counter: 3-array, increment own index only
merging: take max at each position
- commutative, associative, idempotent

summing: add all local counters