RELATIONAL MODEL

The relational model defines a database abstraction based on relations to avoid maintenance overhead.

Key tenets:

→ Store database in simple data structures (relations).
→ Physical storage left up to the DBMS implementation.
→ Access data through high-level language, DBMS figures out best execution strategy.

RELATIONAL MODEL

Structure: The definition of the database's relations and their contents.

Integrity: Ensure the database's contents satisfy constraints.

Manipulation: Programming interface for accessing and modifying a database's contents.

RELATIONAL MODEL

A relation is an unordered set that contain the relationship of attributes that represent entities.

A tuple is a set of attribute values (also known as its domain) in the relation.
→ Values are (normally) atomic/scalar.
→ The special value **NULL** is a member of every domain (if allowed).

Artist(name, year, country)

name	year	country
Wu-Tang Clan	1992	USA
Notorious BIG	1992	USA
GZA	1990	USA

n-ary Relation
=
Table with *n* columns

RELATIONAL MODEL: PRIMARY KEYS

A relation's primary key uniquely identifies a single tuple.

Some DBMSs automatically create an internal primary key if a table does not define one.

Auto-generation of unique integer primary keys:

→ **SEQUENCE** (SQL:2003)
→ **AUTO_INCREMENT** (MySQL)

Artist(name, year, country)

name	year	country
Wu-Tang Clan	1992	USA
Notorious BIG	1992	USA
GZA	1990	USA

RELATIONAL MODEL: PRIMARY KEYS

A relation's primary key uniquely identifies a single tuple.

Some DBMSs automatically create an internal primary key if a table does not define one.

Auto-generation of unique integer primary keys:

→ **SEQUENCE** (SQL:2003)

→ **AUTO_INCREMENT** (MySQL)

Artist(id, name, year, country)

id	name	year	country
123	Wu-Tang Clan	1992	USA
456	Notorious BIG	1992	USA
789	GZA	1990	USA

RELATIONAL MODEL: FOREIGN KEYS

A foreign key specifies that an attribute from one relation has to map to a tuple in another relation.

RELATIONAL MODEL: FOREIGN KEYS

ArtistAlbum(artist_id, album_id)

artist_id	album_id
123	11
123	22
789	22
456	22

Artist(id, name, year, country)

id	name	year	country
123	Wu-Tang Clan	1992	USA
456	Notorious BIG	1992	USA
789	GZA	1990	USA

Album(id, name, year)

id	name	year
11	Enter the Wu-Tang	1993
22	St.Ides Mix Tape	1994
33	Liquid Swords	1995

DATA MANIPULATION LANGUAGES (DML)

Methods to store and retrieve information from a database.

Procedural:

→ The query specifies the (high-level) strategy to find the desired result based on sets / bags.

← Relational Algebra

Non-Procedural (Declarative):

→ The query specifies only what data is wanted and not how to find it.

← Relational Calculus

RELATIONAL ALGEBRA

Fundamental operations to retrieve and manipulate tuples in a relation.
→ Based on set algebra.

Each operator takes one or more relations as its inputs and outputs a new relation.
→ We can "chain" operators together to create more complex operations.

σ	Select
π	Projection
$\cup$	Union
$\cap$	Intersection
$-$	Difference
$\times$	Product
$\bowtie$	Join

RELATIONAL ALGEBRA: SELECT

Choose a subset of the tuples from a relation that satisfies a selection predicate.

→ Predicate acts as a filter to retain only tuples that fulfill its qualifying requirement.

→ Can combine multiple predicates using conjunctions / disjunctions.

Syntax: $\sigma_{\text{predicate}}(R)$

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a2	103
a3	104

$\sigma_{\texttt{a\_id='a2'}}(R)$

a_id	b_id
a2	102
a2	103

$\sigma_{\texttt{a\_id='a2'} \wedge \texttt{ b\_id>102}}(R)$

a_id	b_id
a2	103

```
SELECT * FROM R
 WHERE a_id='a2' AND b_id>102;
```

RELATIONAL ALGEBRA: PROJECTION

Generate a relation with tuples that contains only the specified attributes.
→ Can rearrange attributes' ordering.
→ Can manipulate the values.

Syntax: $\pi_{A1,A2,\ldots,An}(R)$

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a2	103
a3	104

$\Pi_{b\_id-100,a\_id}(\sigma_{a\_id='a2'}(R))$

b_id-100	a_id
2	a2
3	a2

```
SELECT b_id-100, a_id
  FROM R WHERE a_id = 'a2';
```

RELATIONAL ALGEBRA: UNION

Generate a relation that contains all tuples that appear in either only one or both input relations.

Syntax: (R ∪ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

```
(SELECT * FROM R)
UNION ALL
(SELECT * FROM S);
```

(R ∪ S)

a_id	b_id
a1	101
a2	102
a3	103
a3	103
a4	104
a5	105

RELATIONAL ALGEBRA: INTERSECTION

Generate a relation that contains only the tuples that appear in both of the input relations.

Syntax: (R ∩ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R ∩ S)

a_id	b_id
a3	103

```
(SELECT * FROM R)
INTERSECT
(SELECT * FROM S);
```

RELATIONAL ALGEBRA: DIFFERENCE

Generate a relation that contains only the tuples that appear in the first and not the second of the input relations.

Syntax: (R – S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R - S)

a_id	b_id
a1	101
a2	102

```
(SELECT * FROM R)
EXCEPT
(SELECT * FROM S);
```

RELATIONAL ALGEBRA: PRODUCT

Generate a relation that contains all possible combinations of tuples from the input relations.

Syntax: **(R × S)**

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

```
SELECT * FROM R, S;
```

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R × S)

R.a_id	R.b_id	S.a_id	S.b_id
a1	101	a3	103
a1	101	a4	104
a1	101	a5	105
a2	102	a3	103
a2	102	a4	104
a2	102	a5	105
a3	103	a3	103
a3	103	a4	104
a3	103	a5	105

RELATIONAL ALGEBRA: JOIN

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

Syntax: **$(R \bowtie S)$**

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

$(R \bowtie S)$

a_id	b_id
a3	103

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

```
SELECT * FROM R JOIN S USING (a_id, b_id);
```

RELATIONAL ALGEBRA: EXTRA OPERATORS

Rename (ρ)

Assignment (R←S)

Duplicate Elimination (δ)

Aggregation (γ)

Sorting (τ)

Division (R÷S)

OBSERVATION

Relational algebra still defines the high-level steps of how to compute a query.

→ $\sigma_{b\_id=102}(R \bowtie S)$ vs. $(R \bowtie (\sigma_{b\_id=102}(S))$

A better approach is to state the high-level answer that you want the DBMS to compute.

→ Retrieve the joined tuples from **R** and **S** where **b_id** equals 102.

RELATIONAL MODEL: QUERIES

The relational model is independent of any query language implementation.

SQL is the *de facto* standard (many dialects).

```
for line in file.readlines():
  record = parse(line)
  if record[0] == "GZA":
    print(int(record[1]))
```

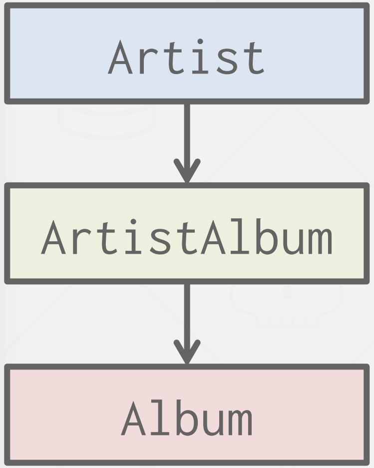
```
SELECT year FROM artists
 WHERE name = 'GZA';
```

DATA MODELS

Relational
Key/Value
Graph
$\boxed{\text{Document / Object}}$ ← Leading Alternative
Wide-Column / Column-family
Array / Matrix / Vectors
Hierarchical
Network
Multi-Value

DOCUMENT DATA MODEL

Embed data hierarchy into a single object.



$\rightarrow R_1$(id,…)

$\bowtie$

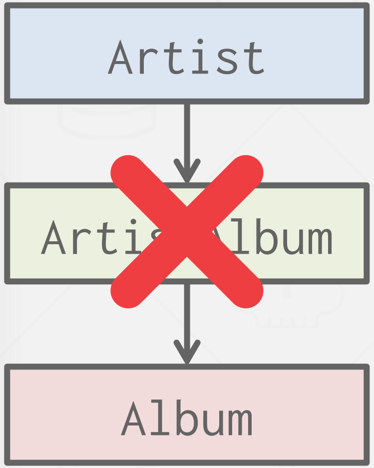
$\rightarrow R_2$(artist_id,album_id)

$\bowtie$

$\rightarrow R_3$(id,…)

DOCUMENT DATA MODEL

Embed data hierarchy into a single object.



$\rightarrow R_1(\texttt{id,…})$

$\bowtie$

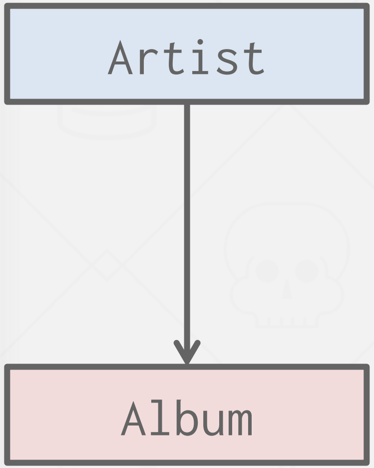
$\rightarrow R_2(\texttt{artist\_id,album\_id})$

$\bowtie$

$\rightarrow R_3(\texttt{id,…})$

DOCUMENT DATA MODEL

Embed data hierarchy into a single object.



Application Code

```
class Artist {
    int id;
    String name;
    int year;
    Album albums[];
}
class Album {
   int id;
   String name;
   int year;
}
```

```
{
 "name": "GZA",
 "year": 1990,
 "albums": [
  {
   "name": "Liquid Swords",
   "year": 1995
  },
  {
   "name": "Beneath the Surface",
   "year": 1999
  }
 ]
}
```

CONCLUSION

Databases are ubiquitous.

Relational algebra defines the primitives for processing queries on a relational database.

We will see relational algebra again when we talk about query optimization + execution.

NEXT CLASS

Modern SQL

→ Make sure you understand basic SQL before the lecture.