Databases

Gaurav Sood

Spring 2015

Data

Data

– Bits on non-volatile storage e.g. magnetic media, SSD

Data

- Bits on non-volatile storage e.g. magnetic media, SSD
- Not just bits on hard disk, but organized bits (a data model)

Data

- Bits on non-volatile storage e.g. magnetic media, SSD
- Not just bits on hard disk, but organized bits (a data model)
  - Nested folders $\sim$ tree

Data

- Bits on non-volatile storage e.g. magnetic media, SSD
- Not just bits on hard disk, but organized bits (a data model)
  - Nested folders $\sim$ tree
  - Rows and columns $\sim$ table

Data Model

– Structure

Data Model

- Structure
  - Rows and columns (Relational Database)

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length
  - Each value in this column has to be X

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length
  - Each value in this column has to be X
  - Child cannot have two parents (tree) (before Gmail Labels)

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length
  - Each value in this column has to be X
  - Child cannot have two parents (tree) (before Gmail Labels)
- (Efficiently Supported) Operations

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length
  - Each value in this column has to be X
  - Child cannot have two parents (tree) (before Gmail Labels)
- (Efficiently Supported) Operations
  - Look up value of key x.

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length
  - Each value in this column has to be X
  - Child cannot have two parents (tree) (before Gmail Labels)
- (Efficiently Supported) Operations
  - Look up value of key x.
  - Find me rows where column 'lastname' is Jordan.

Data Model

- Structure
  - Rows and columns (Relational Database)
  - Key-value pairs (NoSQL systems)
  - Sequence of bytes (Files)
- Constraints
  - All columns must be of same length
  - Each value in this column has to be X
  - Child cannot have two parents (tree) (before Gmail Labels)
- (Efficiently Supported) Operations
  - Look up value of key x.
  - Find me rows where column ‘lastname’ is Jordan.
  - Get next *n* bytes.

Questions to consider

– What are the features of the data? How do we plan to use the data?

Questions to consider

- What are the features of the data? How do we plan to use the data?
- How often are the data updated?

Questions to consider

- What are the features of the data? How do we plan to use the data?
- How often are the data updated?
- What queries are efficiently supported?

Questions to consider

– What are the features of the data? How do we plan to use the data?

– How often are the data updated?

– What queries are efficiently supported?

– How are the data stored on disk – the physical data model?

Questions to consider

- What are the features of the data? How do we plan to use the data?
- How often are the data updated?
- What queries are efficiently supported?
- How are the data stored on disk – the physical data model?
- How are data organized internally – the abstract data model?

What are databases?

– Organized collection of data.

What are databases?

- Organized collection of data.
- Organized to afford efficient retrieval (aim can vary).

What are databases?

- Organized collection of data.
- Organized to afford efficient retrieval (aim can vary).
- Data + Metadata about structure and organization.

What are databases?

- Organized collection of data.
- Organized to afford efficient retrieval (aim can vary).
- Data + Metadata about structure and organization.
- Data that are self-describing $\sim$ have a schema.

What are databases?

- Organized collection of data.
- Organized to afford efficient retrieval (aim can vary).
- Data + Metadata about structure and organization.
- Data that are self-describing $\sim$ have a schema.
- Generally accessed through a Database Management System.

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.
  - **Massive:** Terabytes. Handle data that reside outside memory.

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.
  - **Massive:** Terabytes. Handle data that reside outside memory.
  - **Safe:** Robust to power outages, node failures etc.

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.
  - **Massive:** Terabytes. Handle data that reside outside memory.
  - **Safe:** Robust to power outages, node failures etc.
  - **Multi-user:** Concurrency control. Not one user/ one user per data item. File system concurrency.

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.
    - **Massive:** Terabytes. Handle data that reside outside memory.
    - **Safe:** Robust to power outages, node failures etc.
    - **Multi-user:** Concurrency control. Not one user/ one user per data item. File system concurrency.
    - **Convenient:** High level query languages; declarative: what you want, not describe algorithm

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.
  - **Massive:** Terabytes. Handle data that reside outside memory.
  - **Safe:** Robust to power outages, node failures etc.
  - **Multi-user:** Concurrency control. Not one user/ one user per data item. File system concurrency.
  - **Convenient:** High level query languages; declarative: what you want, not describe algorithm
  - **Efficient:** Performace, performance, performance.

What do Database Systems (DBMS) do?

- Provide efficient, reliable, convenient and safe, multi-user storage of and access to massive amounts of data.
  - **Massive:** Terabytes. Handle data that reside outside memory.
  - **Safe:** Robust to power outages, node failures etc.
  - **Multi-user:** Concurrency control. Not one user/ one user per data item. File system concurrency.
  - **Convenient:** High level query languages; declarative: what you want, not describe algorithm
  - **Efficient:** Performace, performance, performance.
  - **Reliable:** 99.99999% up time.

Key concepts

- Data model
  XML, relational model, etc.

Key concepts

- Data model
  XML, relational model, etc.

- Schema versus data distinction
  Types versus variables
  Schema (types, structure), actual data

Key concepts

- Data model

  XML, relational model, etc.

- Schema versus data distinction

  Types versus variables

  Schema (types, structure), actual data

- Data definition language (DDL)

  For setting up schema

Key concepts

- Data model

  XML, relational model, etc.

- Schema versus data distinction

  Types versus variables

  Schema (types, structure), actual data

- Data definition language (DDL)

  For setting up schema

- Data manipulation or query language (DML)

  For querying and modifying the database

Some Data Models

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format

- Standard for data exchange and representation

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format
- Standard for data exchange and representation
- XML

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format
- Standard for data exchange and representation
- XML
  - Tagged elements

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format
- Standard for data exchange and representation
- XML
  - Tagged elements
  - Nesting of elements

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format

- Standard for data exchange and representation
- XML
  - Tagged elements
  - Nesting of elements
  - Attributes

XML

- Extensive Markup Language

  Similar to HTML

  Tags describe data rather than how to format
- Standard for data exchange and representation
- XML
  - Tagged elements
  - Nesting of elements
  - Attributes
  - Not all elements have attributes

Plain Text

June 5, 2006
Floor Statement of Senator Barack Obama Federal
Marriage Amendment
I agree with most Americans, with Democrats and Republicans, with Vice President Cheney, with over 2,000 religious leaders of all different beliefs, that decisions about marriage, as they always have, should be left to the states.

XML

```
<DOC>
<DOCNO>obama-2006-06-05-01< /DOCNO>
<TEXT>
I agree with most Americans, with Democrats and
Republicans, with Vice President Cheney, with over
2,000 religious leaders of all different beliefs,
that decisions about marriage, as they always have,
should be left to the states.
< /TEXT>
< /DOC>
```

JSON

```
{u’category’: u’Government official’, u’username’:
u’RepJohnLewis’, u’about’: u’Official Congressional
page for Rep. John Lewis’, ... u’name’: u’John
Lewis’, u’hometown’: u’Troy, AL’, ... u’website’:
u’johnlewis.house.gov’, u’phone’: u’(404) 659 -
0116’, u’birthday’: u’02/21/1940’, u’likes’: 62974,
...}
```

Relational Model

– RDBMS– Codd 1970

Relational Model

- RDBMS– Codd 1970
- Database: set of named relations (or tables)

Relational Model

- – RDBMS– Codd 1970
- – Database: set of named relations (or tables)
- – Each relation has a set of named attributes (or columns)

Relational Model

- RDBMS– Codd 1970
- Database: set of named relations (or tables)
- Each relation has a set of named attributes (or columns)
- Each attribute (column) has a type (or domain)

  There is also concept of an enumerated domain

Relational Model

- RDBMS– Codd 1970
- Database: set of named relations (or tables)
- Each relation has a set of named attributes (or columns)
- Each attribute (column) has a type (or domain)

  There is also concept of an enumerated domain
- Each row (= tuple)

Relational Model

- RDBMS– Codd 1970
- Database: set of named relations (or tables)
- Each relation has a set of named attributes (or columns)
- Each attribute (column) has a type (or domain)
  There is also concept of an enumerated domain
- Each row (= tuple)
- Schema– structure, name, type of attributes

Relational Model

- RDBMS– Codd 1970
- Database: set of named relations (or tables)
- Each relation has a set of named attributes (or columns)
- Each attribute (column) has a type (or domain)
  There is also concept of an enumerated domain
- Each row (= tuple)
- Schema– structure, name, type of attributes
- Everything is a table

Relational Model

– Relationships are implicit; no pointers

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.
- Example:

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.
- Example:
  - [course, Student id], [student id, student name]

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.
- Example:
  - [course, Student id], [student id, student name]
  - Just a shared id

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.
- Example:
  - [course, Student id], [student id, student name]
  - Just a shared id
  - Really bad for performance, all student names for a course, look up all the ids

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.
- Example:
  - [course, Student id], [student id, student name]
  - Just a shared id
  - Really bad for performance, all student names for a course, look up all the ids
  - Hierarchical can be quicker if all students in one course

Relational Model

- Relationships are implicit; no pointers
  - Physical Data independence
  - Compare to Network Data Model etc.
- Example:
  - [course, Student id], [student id, student name]
  - Just a shared id
  - Really bad for performance, all student names for a course, look up all the ids
  - Hierarchical can be quicker if all students in one course
  - But to look up all courses student X has taken, just as good (bad)

Relational Vs. XML

	Relational	XML
Structrure	Tables	Hierarchical
Schema	Fixed in advance	Flexible, self-describing
Queries	Easy	Querying not as easy
Ordering	Unordered	Implied order (document)

Algebra of Tables

– In algebra, concept of algebraic optimization – Rules to simply calculation

Algebra of Tables

- In algebra, concept of algebraic optimization – Rules to simply calculation
- SQL uses relational algebra

Algebra of Tables

– In algebra, concept of algebraic optimization – Rules to simply calculation

– SQL uses relational algebra

– All RDBMS optimize relational algebra ‘calculations’

Algebra of Tables

- In algebra, concept of algebraic optimization – Rules to simply calculation
- SQL uses relational algebra
- All RDBMS optimize relational algebra ‘calculations’
- Cost-based optimization

Algebra of Tables

- In algebra, concept of algebraic optimization – Rules to simply calculation
- SQL uses relational algebra
- All RDBMS optimize relational algebra ‘calculations’
- Cost-based optimization
- We can leave this optimization to program

Algebra of Tables

– Basic Relational Algebra

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$
- Extended Relational Algebra:

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$
- Extended Relational Algebra:
  - Duplicate elimination, d

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$
- Extended Relational Algebra:
  - Duplicate elimination, d
  - Grouping and aggregation, g

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$
- Extended Relational Algebra:
  - Duplicate elimination, d
  - Grouping and aggregation, g
  - Sorting, t

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$
- Extended Relational Algebra:
  - Duplicate elimination, d
  - Grouping and aggregation, g
  - Sorting, t
- Sets Vs Bags

Algebra of Tables

- Basic Relational Algebra
    - Union, Intersection, Difference
    - Selection, σ
    - Projection, Π
    - Join, $\bowtie$
- Extended Relational Algebra:
    - Duplicate elimination, d
    - Grouping and aggregation, g
    - Sorting, t
- Sets Vs Bags
    - Sets: {a,b,c} (RA, Papers)

Algebra of Tables

- Basic Relational Algebra
  - Union, Intersection, Difference
  - Selection, σ
  - Projection, Π
  - Join, $\bowtie$
- Extended Relational Algebra:
  - Duplicate elimination, d
  - Grouping and aggregation, g
  - Sorting, t
- Sets Vs Bags
  - Sets: {a,b,c} (RA, Papers)
  - Bags: {a,a,b,c} (SQL)

Select Rows

- Tuples (rows) that satisfy a particular criteria
- $\sigma_{\text{condition}}$Relation
  Pick Certain Rows

Select Rows

- Tuples (rows) that satisfy a particular criteria
- $\sigma_{\text{condition}}$Relation
  Pick Certain Rows
- $\sigma_{GPA>3}$Student
  $\sigma_{GPA>3 \wedge LastName=='Smith'}$Student

Select Rows

- Tuples (rows) that satisfy a particular criteria
- $\sigma_{\text{condition}}$Relation
  Pick Certain Rows
- $\sigma_{GPA>3}$Student
  $\sigma_{GPA>3 \wedge LastName=='Smith'}$Student
- In SQL, selection using ‘Where’ (and project using ‘Select’)

Project

- $\Pi_{A1,\ A2}$Relation
  Pick Certain Columns

Project

- $\Pi_{\text{A1, A2}}$Relation
  Pick Certain Columns
- $\Pi_{\text{SSN, GPA}}$Student

Project

- $\Pi_{\text{A1, A2}}$Relation
  Pick Certain Columns
- $\Pi_{\text{SSN, GPA}}$Student
- Both Select and Project
  $\Pi_{\text{SSN, GPA}}(\sigma_{GPA>3}\text{Student})$
  $\Pi_{\text{A1, A2}}$Expression
  $\sigma_{\text{condition}}$Expression

SQL

```
Select Col1, Col2 (or *)
FROM R1
Where Condition
```

Cross-Product

– Cartesian Product

Cross-Product

- – Cartesian Product
- – R1 X R2; Every tuple in R1 with each tuple in R2

Cross-Product

- Cartesian Product
- R1 X R2; Every tuple in R1 with each tuple in R2
- Size of ouput: R1 * R2

Cross-Product

- Cartesian Product
- R1 X R2; Every tuple in R1 with each tuple in R2
- Size of ouput: R1 * R2
- SELECT *
  FROM R1 CROSS JOIN R2;

Cross-Product

- Cartesian Product
- R1 X R2; Every tuple in R1 with each tuple in R2
- Size of ouput: R1 * R2
- SELECT *
  FROM R1 CROSS JOIN R2;
- SELECT *
  FROM R1, R2;

Natural Join

– Means Equi-Join

Natural Join

- Means Equi-Join
- For every record in R1, find a tuple in R2 that satisfies a particular condition

Natural Join

- Means Equi-Join
- For every record in R1, find a tuple in R2 that satisfies a particular condition
- Select *
  From R1, R2
  Where R1.A = R2.A

Natural Join

- Means Equi-Join
- For every record in R1, find a tuple in R2 that satisfies a particular condition
- Select *
  From R1, R2
  Where R1.A = R2.A
- Or, do the cross-product and then filter

Natural Join

- Means Equi-Join

- For every record in R1, find a tuple in R2 that satisfies a particular condition

- Select *
  From R1, R2
  Where R1.A = R2.A

- Or, do the cross-product and then filter

- Select *
  From R1 Join R2
  On R1.A = R2.B

Theta Join

– Join without the equality condition

Theta Join

– Join without the equality condition

– Equi-Join is a special case of theta join

Theta Join

- Join without the equality condition
- Equi-Join is a special case of theta join
- Find ALL weather stations within 5 miles of centroid of a zip code

Theta Join

- Join without the equality condition
- Equi-Join is a special case of theta join
- Find ALL weather stations within 5 miles of centroid of a zip code
- Select Distinct WStations
  From Wstations h, ZipCode s
  Where distance(h.location, s.location) < 5
- ‘distance’ is a user-defined function

Theta Join

- Join without the equality condition
- Equi-Join is a special case of theta join
- Find ALL weather stations within 5 miles of centroid of a zip code
- Select Distinct WStations
  From Wstations h, ZipCode s
  Where distance(h.location, s.location) < 5
- ‘distance’ is a user-defined function
- Band Joins or Range Joins

Theta Join

– Outer Join

Theta Join

- Outer Join
- All tuples in R1 and pad out other values with NULLs

Theta Join

- – Outer Join
- – All tuples in R1 and pad out other values with NULLs
- – Left outer join

  Select *
  From R1 LEFT OUTER Join R2
  On R1.A = R2.B

Theta Join

- Outer Join
- All tuples in R1 and pad out other values with NULLs
- Left outer join
  Select *
  From R1 LEFT OUTER Join R2
  On R1.A = R2.B
- Right outer join

Theta Join

– Outer Join

– All tuples in R1 and pad out other values with NULLs

– Left outer join
Select *
From R1 LEFT OUTER Join R2
On R1.A = R2.B

– Right outer join

– Full outer join

Union

- R1 U R2
  Removes duplicates by default

Union

- R1 U R2
  Removes duplicates by default
- Select * FROM R1
  UNION
  Select * FROM R2

Union

- R1 U R2
  Removes duplicates by default
- Select * FROM R1
  UNION
  Select * FROM R2
- If you wanted duplicates, you want to do
  ‘UNION ALL’

Difference

– R1 - R2

Difference

- R1 - R2
- Select * From R1
  Except
  Select * From R2

Difference

- R1 - R2

-
  ```
  Select * From R1
  Except
  Select * From R2
  ```
- Looks up all tuples in R1 and takes out tuples that it also sees in R2

  R2 can have other tuples

  Everything in R1, remove things that also appear in R2

Intersection

– Not a fundamental operator

Intersection

- Not a fundamental operator
- R1 Intersect R2 = R1 - (R1 - R2)

Intersection

- Not a fundamental operator
- R1 Intersect R2 = R1 - (R1 - R2)
- Can also express it as join

(Virtual) Views

- Defining and Using (Virtual) Views

  Three-level vision of database

  Physical layer, Conceptual (abstraction of the data, relations), Logical layer (further abstraction,)

  Real applications use lots and lots of views

(Virtual) Views

- Defining and Using (Virtual) Views

  Three-level vision of database

  Physical layer, Conceptual (abstraction of the data, relations), Logical layer (further abstraction,)

  Real applications use lots and lots of views

- Benefit of Views

  Hide some data from some users (Authorization etc.)

  Make certain queries easier, more natural

  Modularity of database access

(Virtual) Views

– Defining and Using (Virtual) Views

Three-level vision of database

Physical layer, Conceptual (abstraction of the data, relations), Logical layer (further abstraction,)

Real applications use lots and lots of views

– Benefit of Views

Hide some data from some users (Authorization etc.)

Make certain queries easier, more natural

Modularity of database access

– Views

Query over relations

View V = ViewQuery(R1, R2,... RN)

Schema of V is schema of query result

Query over views

– V can be thought of as a table

Query over views

- V can be thought of as a table
- Q is actually rewritten in terms of R1, R2, ...RN

Query over views

- V can be thought of as a table
- Q is actually rewritten in terms of R1, R2, ...RN
- R1 can also be view

Query over views

- V can be thought of as a table
- Q is actually rewritten in terms of R1, R2, ...RN
- R1 can also be view
- CREATE View VName (A1, ..., AN) As Query

Query over views

- V can be thought of as a table
- Q is actually rewritten in terms of R1, R2, ...RN
- R1 can also be view
- CREATE View VName (A1, ..., AN) As Query
- Just define the view and use them (System optimizes in terms of ‘base tables’)

Query over views

- V can be thought of as a table
- Q is actually rewritten in terms of R1, R2, ...RN
- R1 can also be view
- CREATE View VName (A1, ..., AN) As Query
- Just define the view and use them (System optimizes in terms of ‘base tables’)
- View contents not stored

Query over views

- V can be thought of as a table
- Q is actually rewritten in terms of R1, R2, ...RN
- R1 can also be view
- CREATE View VName (A1, ..., AN) As Query
- Just define the view and use them (System optimizes in terms of ‘base tables’)
- View contents not stored
- A convenient view is a join

Modifying Views

– We cannot insert, update, delete as V isn't stored

Modifying Views

– We cannot insert, update, delete as V isn't stored

– To alter V, we have to modify the base tables

Materialized Views

– Creates a physical table

Materialized Views

- – Creates a physical table
- – But . . .

Materialized Views

- Creates a physical table
- But . . .
  - Tables can be very large

Materialized Views

- Creates a physical table
- But . . .
  - Tables can be very large
  - What happens when modifications happen to the base tables

Materialized Views

- Creates a physical table
- But . . .
  - Tables can be very large
  - What happens when modifications happen to the base tables
  - Modification to base data can invalidate the view

SQLite: Data Types

– NULL – The value is a NULL value

SQLite: Data Types

– NULL – The value is a NULL value

– INTEGER – a signed integer

SQLite: Data Types

- NULL – The value is a NULL value
- INTEGER – a signed integer
- REAL – a floating point value

SQLite: Data Types

- NULL – The value is a NULL value
- INTEGER – a signed integer
- REAL – a floating point value
- TEXT – a text string

SQLite: Data Types

- NULL – The value is a NULL value
- INTEGER – a signed integer
- REAL – a floating point value
- TEXT – a text string
- BLOB – a blob of data