Databases

Kevalee Shah

January 9, 2019

Contents

1 General Notes

- Interface represents abstraction used by applications
- However can lose efficiency from abstraction
- DBMS - database management system, does ACID and CRUD
- CRUD - Create, Read, Update, Delete
- ACID transactions - atomicity, consistency, isolation, durability
- Redundant data - data is redundant if it can be deleted and then reconstructed from the data remaining
- Tradeoff with redundant data - if many updates need ot be done then redundancy is an issue, however if mainly for querying then high redundancy can actually increase speed

2 Entity Relationship Diagrams

- a model to show entities, attributes and relationships
- **Entities** - *square* to show nouns, e.g. movie, person
- **Attribute** - *oval* to show properties, e.g. id, genre, year

  key is underlinesd - uniquely identifies the entity instance

  relationships can have atrributes - e.g. actsin can have position/character name
- **Relationships** - *diamonds* to show verbs, e.g. directs, acts in
- When designing need ot consider the scope of the data model
- Weak Entity - when the existence of an entity depends on another entity it is weak - e.g. movie release depends on movie

  The relationship between them is called the **identifying relationship**

2.1 Cardinality of a relation

- One-to-many: Each T can only be related to one S e.g. Student can only have one college
- Many-to-one: Each S can only be realted to one T e.g. Colleges can have many students
- One-to-one: when R is both One-to-many and Many-to-one
- Many-to-many

3 Relational Algebra

- For set S, T, $S \times T = \{(s, t) \mid s \in S, t \in T\}$

  e.g. $S = \{1, 2, 3\}$, $T = \{a, b, c\}$, $S \times T = \{(1, a), (2, b), (3, c)\}$
- A relation over $S \times T$ is set R where $R \subseteq S \times T$
- S, T are domains, R needs to be finite to be stored
- n-ary relations when there are n domains
- Stored as tuple $(A_1, s_1), (A_2, s_2), \ldots, (A_n, s_n)$, where A_i represents the name of the domain and s_i is a value in the domain S_i
- Schema is the structure of the database e.g Sudents(name:string, sid:string, age:int)

- Queries must be well formed - all columns of result are distinct: $Q_1 \times Q_2$ columns must be distinct, but for $Q_1 \cup Q_2$ columns must be the same
- Types of queries:

  selection

  projection

  product

  difference

  union

  intersection

  renaming

- Join notation: Given $R(\mathbf{A, B})$ and $S(\mathbf{B, C})$

$$R \bowtie S = \{t | \exists u \in R, v \in S, u.[\mathbf{B}] = v.[\mathbf{B}] \wedge t = u.[\mathbf{A}] \cup u.[\mathbf{B}] \cup v.[\mathbf{C}]\}$$

  (t for tuple)
  This gives where there is the same B in R and S, it forms a table with the B as well as the A and C

4 Redundant Data

Issues:

- Insertion - if everything is in one table, then you can fill in partial information. Therefore a reason why redundant data is needed. E.g. we might was to add a person without knowing their role, or want to add a movie without knowing its release date
- Deletion - if a lot of redundant data, when you want to delete an entry need to delete it from many places
- Update - if a lot of redundant data, need to update at many places

Therefore really depends on whether database is mainly for querying or for updating and how many concurrent updates are happening - all affect throughput

5 Keys

- Suppose we have $R(\underline{\mathbf{Z}}, Y)$ and $S(\mathbf{W})$. If $\mathbf{Z} \subseteq \mathbf{W}$ then ***Z*** *is a foreign key in* ***S*** *for R*
- A databse is said to have **Referential Integrity** if all foreign key constraints are satisfied

5.1 Multiple Relationships

Suppose we have two many-many relationships: $R(\underline{\mathbf{X, Z}}, \mathbf{U})$ and $Q(\underline{\mathbf{X, Z}}, \mathbf{V})$
We don't have to use 2 tables for this but can merge them into one table using a type column e.g. $RQ(X, Z, type, U, V)$ where $type = \{\mathbf{r,q}\}$ - when $type = \mathbf{r}$ we know $V = NULL$

6 Index

Bruteforce approach to JOINS:

```
scan (a,b) in R
scan (b', c) in S
if b = b' then create (a,b,c)
```

To prevent having to scan S as well, can use **indexes** - a datastructure created by the database that reduces the time to locate records
TRADEOFF: indexes can increase speed of reads, but slows down updates

7 SQL

- Based on multisets not set:

  This means `select B, C from R` will give all of B and C, including repeats. To get rid of repeats use `distinct`. Using multisets is important as needed for calculations such as `mean, min, max`

- `NULL` is a place holder not a value

  e.g. Three way logic truth table

$\wedge$	T	F	N
T	T	F	N
F	F	F	F
N	N	F	N

- annoying results such as cannot test if a value is `NULL` as `NULL = NULL` is `NULL`

8 Bacon Number

8.1 Compositions

Let $R \subseteq S \times T$ and $Q \subseteq T \times U$

$$Q \circ R \equiv \{(s, u) | \, \exists t \in T, (s, t) \in R \wedge (t, u) \in Q\}$$

- e.g. if $A = \{(A, B), (A, D), (B, C), (C, C)\}$ then $A \circ A = \{(A, C), (B, C), (C, C)\}$
- Composition in terms of partial functions:

  $f \in S \to T \qquad g \in T \to U$
  Is defined by: $(g \circ f)S = g(f(s))$

9 Graphs

- $G = (V, A)$

  Directed graph with V nodes and a binary relation A over V

  $A \subseteq V \times V$

- If R is a binary relation over S such that $R \subseteq S \times S$ then an **iterated composition** is defined as:
  $R^1 = R$
  $R^{n+1} = R \circ R^n$
- If $G = (V, A)$ is a directed graph and $(u, v) \in A^k$ then there is at least one path in G from $u \to v$ of length k

9.1 R-Distance

- Suppose $s_0 \in \pi_1(R)$

  where π notation means projection - selecting s_1 from R

  This means that there is a pair $(s_0, s_1) \in R$
- For $(s_0, s_1) \in R$, distance from s_0 to s_1 is 1
- For any other $s' \in \pi_2(R)$ then the distance from s_0 to s' is the least n such that $(s_0, s') \in R^n$

- To get bacon numbers, keep on having to join bacon numbers - but we dont know when to stop

9.2 Transitive Closure

R is a binary relation over S such that $R \subseteq S \times S$
The **transitive closure** of R is denoted R^+, and it is the smallest binary relation on S such that $R \subseteq R^+$ and R^+ is **transitive**:

$$(x, y) \in R^+ \wedge (y, z) \in R^+ \to (x, z) \in R^+$$

Then

$$R^+ = \bigcup_{n \in 1,2,\ldots} R^n$$

As relations are finite there is some n, but we cannot compute n unless we know the contents of R and therefore in SQL cannot compute the transitive closure - a motivation for graph orientated databases

10 Neo4j

- nodes and binary relationships between nodes
- nodes and relationships can have attributes - properties
- Cypher is the query language
- code for bacon numbers:

```
MATCH (p:Person)
where p.name <> "Kevin Bacon
with p
match paths=allshortestpaths (
(m:Perons{name: "Kevin Bacon"} )
-[:ACTS_IN*]- (n:Person {name: p.name} ) )
return distinct p.name,
length(paths)/ 2 as bacon_number
order by bacon_number desc;
```

11 Graph-oriented database

These are good for when your data is seldom updated, but often read.
e.g. can have a database optimised for updates, and then every so often you extract info for the read-optimized database. Queries are done on the read-optimized database

- Stores data in a semi-structured form
- Disadvantages of big table:

  - could speed up some queries
  - but if you're mainly using 1/2 keys, then the database has to scan a big table to get info to gather - will still take time
  - Indexes might help but still not ideal
  - Want a type of database where you get all data associated with a certain key
- results in a lot of redundant data, but very fast reading

11.1 Key-Value Store

- Map a key to a block of bytes
- intepretation of the block of bytes is left for the application

12 OLAP VS OLTP

- **OLAP** - Online Analytical Processing

  For analysis, historical data, optimized for reading, high data redundancy and very very big database
- **OLTP** - Online Transactional Processing

  For day to day operations, current data, optimized for updating, low data redundancy and big database
- Operational Database $\underset{\rightarrow}{ETL}$ Data Warehouse

  Operational for updates

  ETL - extract, transform, load

  Data Warehouse - business analysis queries

13 Data Cube

- Data modeled as $n-$ dimensional
- Each dimension has hierarchy
- Each point records a fact
- Easy to aggregate and cross-tabulate across dimensions

14 Column vs Row oriented

Row: easy to add or modify, but might read unecesary data
Column: only need to read relevant data, hard to write, column mostly for read-intensive databases