Databases

CLASS. 1

19/4/2

--sounds complicated, but is very intuitive

--just a way of storing information

-large amounts

-easily accessed

-easily managed

-easily updated

--special type of data structure

Table Analogy

[Students]

-------------------------------

Name | SID | year | major

Joe 25 1 CS

Mary 47 2 English

-------------------------------

DBMS(database Management System)

* When people say DB they often mean DBMS
* Software that interacts with the DB data structure , in a user friendly way
* Allow creation, modification , administration, etc…
* EX: MySQL, PostyleSQL, Create, Access

“Relational” DB

-distinction SLW relational DB and Non-relational

-not referring to “relationships”

-technical term from relational algebra related to set theory

-“relation” is just a set of tuples finite, ordered list of elements

-every type has the same # of elements and every elements in the same position is a member of the same domain

-“domain”

-a set of every possible value that a given element of a type could have

- in the table analogy

Type = row

Attribute/element = column

Domain = data type(int, string, Boolean, etc)

(defined by)

Relation = table

-example of relational DBs:

OracleDB , MySQL, DcstyreSQL, SQLite, etc..

-example of non-relational DBs

-key-value stores: Radis

-document store: MongoDB

Schema

-structure of a DB

-map of haw its constructed and haw things relate to each other

-some schema are static (non- changing) and some are dynamic

Query

-a single action taken on the database

-tried information

-updating or creating data(or deleting)

-creating or changing the schema

-looks like a line of code

Transaction

-sequence of queries run against , DB

-single unit of work (dependent on each other)

Ex:

-----------------------

Name | money

Bob $40

Alice $80

------------------------

Bob gives Alice $20

-subtract 20 from Bob

-Add 20 to Alice

ACID(Atomicity Consistency, Isolation, Durability)

-Transactions must demonstrate all 4 to be ACID – compliant

Atomicity

-everything succeeds or fail together

Consistency

-all queries leave the DB in a consistent state

Isolation

-no transaction interferes with another transaction

Durability

-changes made by a transaction are available in the DB in case of system failure

SQL (Structured Query Language)

-a programming language basial on relational algebra

-used to communication with a DBMS  
 -this a specific abstraction used in specific DBMS’s (mySQL,SQLite, DcstyreSQL,…)

Normalization（dc-）

-Process of structing the schema to improve :

-reducing redundancy

-data integrity(consistent, up-to-data, acoucate)

-efficiency

----------------------------

Author | ISBN

King, Steven | 10234, 24345, 38038

Fields, | 13413,1341

Was the book #38038 written by Steven King?

-----------------------------------------------

Author | ISBN

King, Steven | 10234

King, Steven | 13413

King, Steven | 24345

King, Steven | 38038

-denormalization

-adding redundancy for some effect

-spending up queries(efficiency)

ORM( Object- relational Models)

-libraries or packages that abstract DB

interaction away from the DBMS languages(SQL)

into a more familiar language

-lets you continue coding in your language of choice

-there are inevitably many ORMs for any given DBMS <-------> programming language pair

-in the class we will look at:

MySQL <-----------> Python

MySQL <-----------> Ruby

I/R Model( Entity- Relationship , Model)

* Entities: things that exist and are distinguishable

Ex: Bob, Alice, book, place, tree, job, coupon code

-relationships: two or more entities participate in a relationship

Ex: owes money, wrote book, lives at, friends

Student ------- attends ----------- class

|

Registered

Entities are represented by rectangles



relationship are represented by rectangles



Entities participate in relationships via lines

----



Attributes of entities represented bu ellipses



Ex:

(time CRN Capacity) Class---------attend --------Student (name SID year major)

| \ |

Teacher held-in ~~site in~~

| \ |

(name FID department)Teacher Classroom (building room)

-For any real world domain, we could model a potentially infinite number of relationships. But we really don’t want to

-ideally we model the minimum relationships necessary to answer whatever problem were trying to solve

CLASS. 2

19/4/4

DATABASE DESIGN

-ACTIVITY OF DEVELOPING A SCHEMA FOR A DETABASE FOR A GIVEN DATABASE

Info

|

Conceptual data model(managers, etc.)

|

Logical data model(developers?)

|

Physical data model (database admins)

Conceptual Data Model

-for removed from the actual DB implementation

-just to help think about the data

-identify objects and relationship between them.

-the goal of this step is to create a complete descriptor of the database in an informal paper-anti-pencil formal

Logical Data Model

-start to transform the conceptual design into a more formal schema (logical schema)

Physical Data Model

-start modeling the really of the DB implementation

-choose a DBMS



-choose data types

-server design/ network design



-“choosing access methods” -> choose a DBMS

Why each level?

Conceptual

-helping you think/ talk about data

-semantic clarity

Logical

-help in determine what DBMS would be best

-at this stage you can start to structure the schema more abstractly to avoid redundancy and to improve efficiency(normalization)

Physical

-we want a real DB

-performance, optimization (de-normalization)

CONCEPTUAL DESIGN

The consensus choice for conceptual modelling is the Entity-Relationship (ER) model

Two primitives

Entity things that exist and are distinguishable

Eg: Bib Alice Susan ‘car DU

Relationships : two or more entities can participate in a relationship

Eg: Bob bar roused. Susan’s car Susan “is a patron of “ DU

Bath Entities and Relationships can have attributes

Eg: Bob has an age; Susan’s car has color Susanis a $1000 patron of DU.

Take a group of all entities with the same attributes, that forms an “Entity set”

Eg: all person, all cars, all universities

Relationships are also grouped into homogenous sets called “Relationship Sets”. All the pairs of Entities such that the relationship is true for them.

Eg: All the (person, car) pairs such that the person owns the car.

The overall structure of a DB can be expressed graphically as an E-e diagram, typically with the following components:

Entity sets ->



Relationship sets ->



Attributes ->



Links between ->



Ex: we want a library DB where authors have written books about various subjects (one author per book). It also has info about Libranes that carry books on the various subjects.

title

/

Library. ---- Carries ---- book.

Author(name , SSN, citizenship). (writes)

|

~~Lsrite on~~

|

Subject (subject) ------- index ------------ book. (title, ISBN)

|

~~carry~~

|

Library (name, city )

Q: I have an author “Isaac Asimov”, which books has he written?

Author (name , SSN, citizenship)

|

write

|

(tile, ISDN) Book (tile ISDN) ------- index ----------- Subject (name desc)

|

Carry (copies) (duration)

|

Library (name, address)

Relationship is an ordered set of Entities.

Relationship set is a set of ordered sets of Entities.

‘borrowed’ -> {(Bob, Susan’s car)

(Alice, Bob’s book),

(Joe, John’s nail clippers)}

A relationship might involve;

2 entities: bring relationship

3 entities: ternary relationship

N entities: N-ary relationship

A binary relationship between two sets of entities may be:



1 : 1 relationship



Eg: a person haring a SSN



Person {…………………}

\ / \ //// \\



SSN {…………………}



1 : N relationship

Eg: father having children

Father {……………….}

/\ /\ ///\\

Child {……………….}

M : N Relationship

Eg: students enrolled in closes

Student {……………………}

/\ X /\ |///\\

Class {……………………}

Participation Constraints

-addresses whether or not Entities can or must participate in some relationship

E1. =========== R ============ E2

Total partial

Total participation constraint

-every entity in E1 must participate in the relationship R

Partial participation constraint

-each entity in E2 may or may not participate in the relationship R

Keys

Keys are a way of distinguishing Entities or Relationships, and there are various kinds of keys with ranging specifically

‘Super key’

– a set of one or more attributes of an entity or relationship whose value uniquely identify that entity or relationship

Eg: the name “Isaac Asimov” and a particular SSN and a citizenship

‘candidate key’

– a minimal super key, i.e., one where no subset of its attributes or a super key

Eg: just the SSN

‘Primary Key’

-One specific candidate key chosen to serve as the identifier for a relationship or entity set.

‘Foreign Key’

-A set of one or more attribute of an entity or relationship that serves as a primary key for another entity or relationship

Eg: for student enrolling in a class, we might use student ID as a foreign key in the enroll relationship.