CS-322: Introduction to Database Systems

March 13, 2017

Contents

1 Administrative stuff **2**
- 1.1 Reading 2
- 1.2 Grading 2

2 Lecture 1 **2**
- 2.1 Data vs Infororg-caldavmation 2
- 2.2 Look up house of cards big data here 3
- 2.3 Look up emc report for 2020 3
- 2.4 Database Management System (DBMS) 3
- 2.5 Database 3
- 2.6 DBMS Design and Architecture 4

3 Lecture 2 **4**
- 3.1 Data model 4
- 3.2 Conceptual design 5
- 3.3 Entity-Relationships Model Basics 5
- 3.4 Constraints 6
- 3.5 Complex relationships 8
- 3.6 Conceptual Design 9

4 Lecture 3 **9**
- 4.1 Role of database system 9
- 4.2 What is a data model? 10
- 4.3 Relation Model: Structural aspect 11
- 4.4 Relational Model: Integrity Aspect 12
- 4.5 Domain Constraints 12
- 4.6 Relational Model: Keys 13
- 4.7 Relational Model: Foreign Keys 13

4.8 Mapping ER with key constraints 14
4.9 Translating Weak Entity Sets 16
4.10 Translating ISA Hierarchies to Relations 16
4.11 NoSQL 17

5 Lecture 4 17
5.1 Relational Query Languages 17
5.2 Relational Algebra 18
5.3 Relational Calculus 19

1 Administrative stuff

1.1 Reading

- Book
- Papers

1.2 Grading

- Midterm 30%
- Project 30%
- Final 40% (During summer exam session)

2 Lecture 1

2.1 Data vs Infororg-caldavmation

Data:

- Facts

org-caldav- Basis for reasoning,...

- Can be useful/reorg-caldavlevant or not
- Must be processed

When organized/processed, becomes Information:

- Meaningful
- Relevant
- **Actionable** leads to a solution

2.2 Look up house of cards big data here

2.3 Look up emc report for 2020

2.4 Database Management System (DBMS)

- DBMS is a software to **store**, **manager** and **facilitate access** to databases.
- DBMS = Interrelated data (database) + set of program + se;; refile setup ;;

  ;; targets include this file and any file contributing to the agenda, up to 9 levels deep (setq org-refile-targets (quote ((nil :maxlevel . 9)

(org-agenda-files :maxlevel . 9))))t of p(with-eval-after-load 'orgrograms to access it (software).

- DBMS provides **efficient** (thousands of queries/update per second), **reliable** (availability), **convenient** (physical data indep, high level query languages) and **safe** (protect data from failures: h/w,s/w,power, malicious users) **multi-user** (concu(cfw:open-org-calendar)rrency) storage of and access to **massive** (extremly large TB everyday) amounts of **persistent** (data outlives programs that operate on it) data.

2.5 Database

- A **large**, **integrated**, **structured collection** of data.
- Usually intended to model some real-world enterprise.
- Entities and Relationships.

2.5.1 Is WWW a DBMS?

- Sophisticated search available
- But not really DBMS, data unstructured, "correct answer" not well defined, few guarantees.

2.5.2 Is FS a DBMS?

No, no guarantees when conflict (two edit at same time) or when power out…

2.6 DBMS Design and Architecture

2.6.1 Describing Data

- **Data model** is a collection of concepts for describing data
- **Relation data model**, relation (table with row and columns), schema (structure (columns) of a relation)
- **Nested data model**, not all data fits naturally in tables
- **Schema vs. Data**, type vs variable

2.6.2 Logical and physical data independence

- **Data independence** is the ability to change the

schema at one level of the database system without changing the schema at the next higher level

- **Logical data independence** is the capacity to change

the conceptual schema without changing the user views

- **Physical data independence** is the capacity to

change the internal schema without having to change the conceptual schema or user views ⇒ allows for modularity

3 Lecture 2

3.1 Data model

- A **data model** is a collection of concepts for describing data
  - High-level : hides lot of low-level storage details
  - Relational, XML, Graph, Object-Oriented,...
- **Relational data model**
  - Set of records
  - **Relation**: Table with rows and columns
  - **Schema**: Describes the structure (columns) of a relation
- **Schema vs. Data**

  - Type vs. variable
  - Description of a particular collection of data, using a given data model

3.2 Conceptual design

- What are the **entities** and **relationships**?
- What info should be stored?
- What are the **integrity constraints**?

3.3 Entity-Relationships Model Basics

3.3.1 Entity

- Real-world object distinguishable from other objects.
- An entity is described (in DB) using a set of attributes

3.3.2 Entity Set

- A collection of similar entities. E.g., all employees
- All entities in an entity set have the same set of attributes. (Until we consider hierarchies, anyway!)
- Each entity set has a **key** (underlined)
- Each attribute has a **domain**

3.3.3 Relationship

- Association among two or more entities. E.g., Fred works in Pharmacy

department

- Can have their own attributes

3.3.4 Relationship Set

A collection of similar relationships. E.g., all employees

3.4 Constraints

Limits the freedom of the data

3.4.1 Key Constraints

- An employee can work in many departments; a department can have many employees: **Many-to-Many**
- In contrast, each department has at most one manager, according to the key constraint on Manages: **One-to-Many**
- Each driver can drive at most one vehicle and each vehicle will have at most one driver: **One-to-One**

/!\ **At most**, i.e. $>= 1$

3.4.2 Participation Constraints

- Every Employee should work in at least one department
- Every Department should have at least one employee

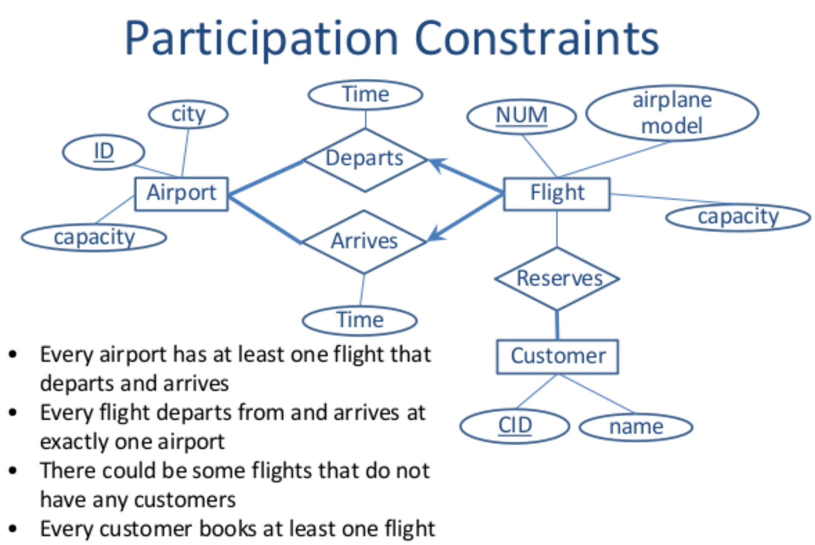
⇒ **Total participation**

- There could be some Employees who are not managers
- Every Department should have at least one manager

If also at most, then it means: "**exactly one**"

- There could be some Customers who do not buy any Products
- There could be some Products which are not bought by any Customer

⇒ **Partial Participation**

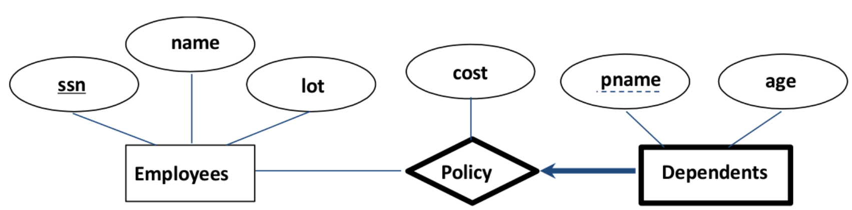


3.4.3 Weak Entities

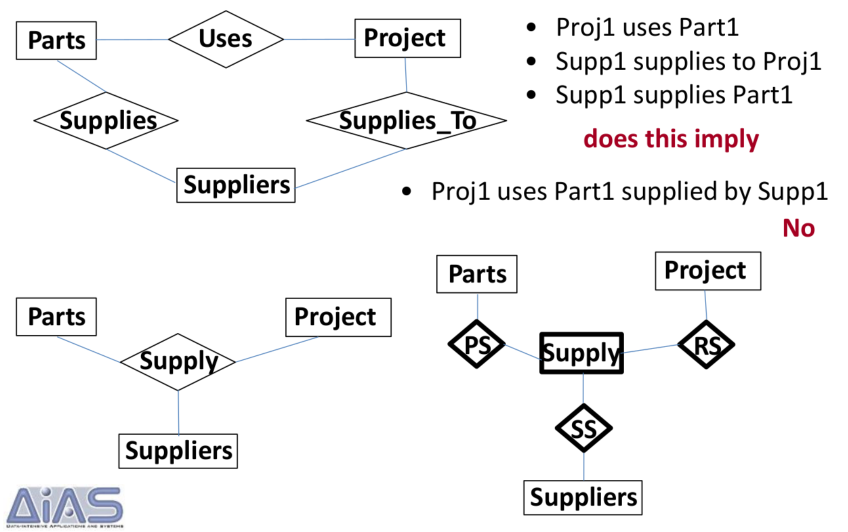
A weak entity can be identified uniquely only by considering the primary key of another (owner) entity

- Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities)
- Weak entity set must have total participation in this identifying relationship set

Weak entities have only a "partial key" (dashed underline)



3.4.4 Ternary Relationships



- Proj1 uses Part1
- Supp1 supplies to Proj1
- Supp1 supplies Part1

does this imply

- Proj1 uses Part1 supplied by Supp1

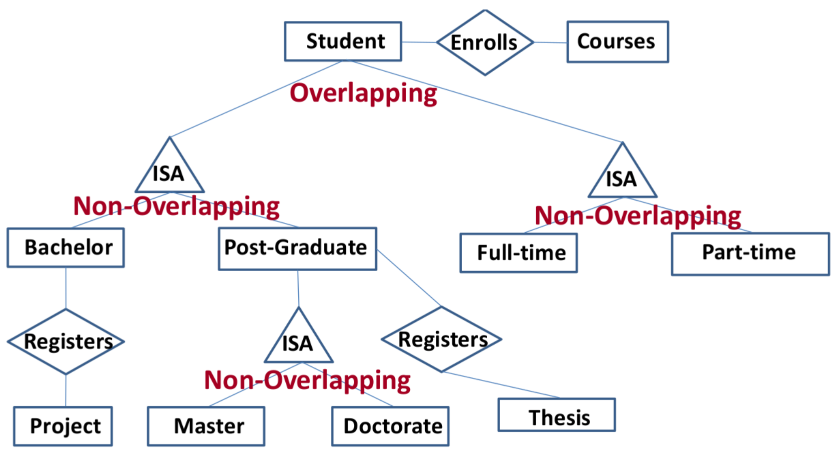
$\Rightarrow$ No
Can be done for n-ary relationships

3.5 Complex relationships

3.5.1 ISA ('is a') Hierarchies

- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.
- **Overlap constraints**: Can Joe be an HourlyEmps as well as a ContractEmps entity? (Allowed/Disallowed)

- **Covering constraints**: Does every Employees entity also have to be an HourlyEmps or a ContractEmps entity? (Yes/No)
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass. (i.e., not appropriate for all entities in the superclass)
  - To identify entities that participate in a particular relationship (i.e., not all superclass entities participate)



3.5.2 Aggregation

- Used to model a relationship involving a relationship set
- Allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships

3.6 Conceptual Design

4 Lecture 3

4.1 Role of database system

- Database: integrated, shared data collection
- Integrated
  - Eliminate needless redundancy

- Maintain strong consistency
- Shared
  - Application written by programmers in multiple languages
  - End-users who use applications, forms, CLI to interact
- Database systems shield users from
  - How data is stored (bits & bytes, 1 vs N files, 1 vs N disks…)
  - How data is accessed (btree, hashtable, scan, …)

4.2 What is a data model?

- Collection of application-visible constructs
  - Describe data in application & storage agnostic way
- Constructs to describe structural aspects
  - How do applications perceive the data?
  - Ex: table, graph, associative array…
- Constructs to describe manipulation aspects
  - What operators can applications use?
  - Ex: join, traverse, lookup…
- Constructs to describe data integrity aspects
  - How do we ensure that data manipulation is "correct"?

../../org/school/2016-2017/secondSemester/files/Lecture 3/screenshot_2017-03-06_10-

4.3 Relation Model: Structural aspect

- Database = set of named **relations** (or **tables**)
- Each relation has a set of named **attributes** (or **columns**)
- Each **tuple** (or **row**) has a value for each attribute
- Each attribute has a **type** (or **domain**)
  - integer, real, string, file formats (jpeg,. . .), enumerated and many more
- **Relation Schema**: relation name + field names + field domains
  - Students(sid: string, name: string, login: string, age: integer, gpa: real)

- **Relation Instance**: contents at a given point in time
  - set of rows or tuples. (all rows are distinct with no specific ordering)
  - Cardinality: # rows, Arity or degree: # attributes
- **Database Schema**: collection of relation schemas
- **Database Instance**: collection of relation instances

4.4 Relational Model: Integrity Aspect

- Relational model provides **Integrity Constraints**
  - condition specified on schema that restricts the data that can be stored in any instance
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A **legal** instance of a relation is one that satisfies all specified ICs
  - DBMS should not allow illegal instances.
- With ICs, stored data is more faithful to real-world meaning
  - Avoids data entry errors, too!

4.5 Domain Constraints

- Domain constraints: type of Integrity Constraints
  - Domain specified in schema restricts the data that can be stored in that field
- Enforced by the DBMS whenever tuples are added or modified.
  - Similar to type checks in programming languages

4.6 Relational Model: Keys

- Attribute whose value is **unique** in each tuple
- Or set of attributes whose combined values are unique
- Keys specify **key constraint**
  - Enforced when tuples are inserted/updated
- Key
  - Set of attributes which uniquely identify a tuple
- Candidate Keys
  - If there are multiple keys, each of them is referred to as a candidate key
  - UNIQUE(licence#)
- Primary Key
  - One of the candidate keys is chosen (by DBA)
  - PRIMARY KEY(sid)
- Superkey
  - Superset that includes a key
  - no two distinct tuples can have same values in all key fields

Key must be assigned carefully $\Rightarrow$ must be specific

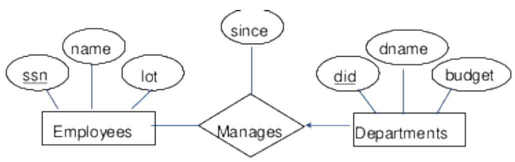
4.7 Relational Model: Foreign Keys

- Set of fields in one relation that 'refer' to a tuple in another relation (like a pointer)
- Foreign keys specify Foreign Key Constraint
  - FK must correspond to the primary key of the other relation
- If all foreign key constraints are enforced, referential integrity is achieved (i.e., no dangling references.)
- FOREIGN KEY (sid) REFERENCES Students(sid)

4.7.1 Enforcing Referential Integrity

- What if an Enrolled tuple with a non-existent student id is inserted? (Reject it!)
- What if a Students tuple is deleted?
  - Also delete all Enrolled tuples that refer to it?
  - Disallow deletion of a Students tuple that is referred to?
  - Set sid in Enrolled tuples that refer to it to a default sid?
  - Set sid in Enrolled tuples that refer to it to a special value null, denoting 'unknown ' or 'inapplicable ' .
- Can specify action taken on violation in SQL

4.8 Mapping ER with key constraints



- Since each department has a unique manager, we could instead combine Manages and Departments.

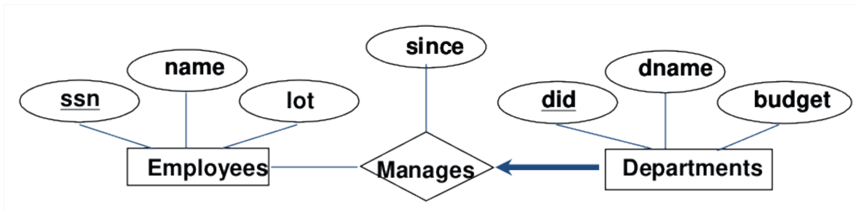
```
CREATE TABLE  Manages(
 ssn  CHAR(11),
 did  INTEGER,
 since  DATE,
 PRIMARY KEY  (did),
 FOREIGN KEY (ssn)   REFERENCES
Employees,
   FOREIGN KEY (did) REFERENCES
Departments)
```

Vs.

```
CREATE TABLE  Dept_Mgr(
 did  INTEGER,
 dname  CHAR(20),
 budget  REAL,
 ssn  CHAR(11),
 since  DATE,
 PRIMARY KEY  (did),
 FOREIGN KEY (ssn)
  REFERENCES Employees)
```

If no manager yet, could have a null ssn foreign key, maybe not what we want...

- Does every department have a manager?
  - If so, this is a participation constraint: the participation of Departments in Manages is said to be total.
  - Every did value in Departments relation must appear in a row of the Manages relation (with a non-null ssn value!)



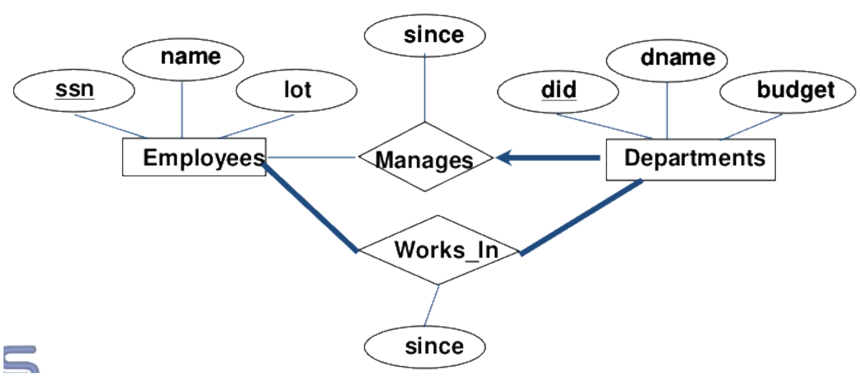

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11) NOT NULL,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees, ON DELETE NO ACTION)
```

This way, department must have a manager

4.8.1 Participation Constraints in SQL: Issues

We cannot capture all participation constraints with PK, FK, not null

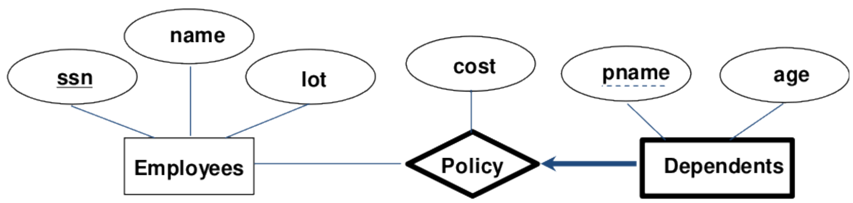
- Ex: Works$_{\text{In}}$ relationship - total participation, no key constraint
- Every did must must appear in a tuple in Works$_{\text{in}}$ table
- Cannot make did foreign key as did is not candidate key in Works$_{\text{In}}$



4.9 Translating Weak Entity Sets

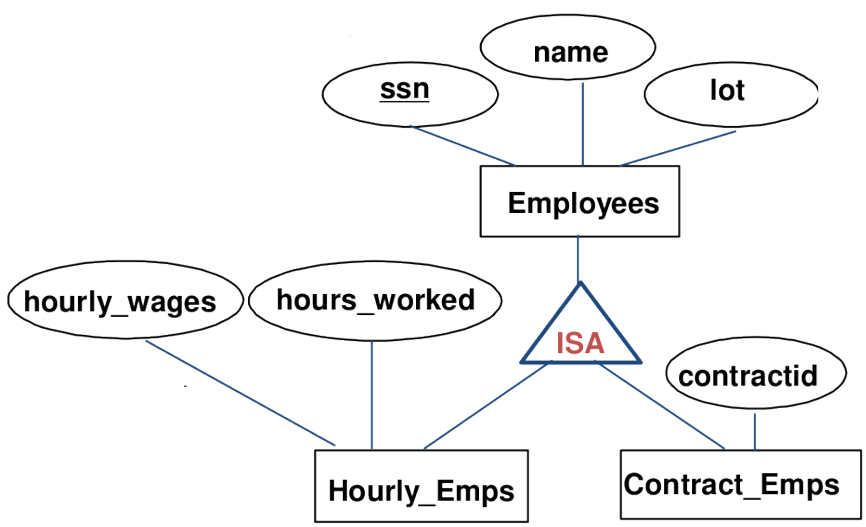
Weak entity set and identifying relationship set are translated into a single table.

- When the owner entity is deleted, all owned weak entities must also be deleted.




```
CREATE TABLE Dep_Policy (
  pname CHAR(20),
  age INTEGER,
  cost REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (pname, ssn),
  FOREIGN KEY (ssn) REFERENCES Employees, ON DELETE CASCADE)
```

4.10 Translating ISA Hierarchies to Relations



4.10.1 Recall

If we declare A ISA B, every A entity is also considered to be a B entity

- Overlap constraints: Can Joe be an $\text{Hourly}_{\text{Emps}}$ as well as a $\text{Contract}_{\text{Emps}}$ entity? (Allowed/disallowed)
- Covering constraints: Does every Employees entity also have to be an $\text{Hourly}_{\text{Emps}}$ or a $\text{Contract}_{\text{Emps}}$ entity? (Yes/no)

4.10.2 General approach

3 relations: Employees, $\text{Hourly}_{\text{Emps}}$ and $\text{Contract}_{\text{Emps}}$.

- Every employee is recorded in Employees.
- $\text{Hourly}_{\text{Emps}}$: For hourly emps, extra info recorded in $\text{Hourly}_{\text{Emps}}$ ($\text{hourly}_{\text{wages}}$, $\text{hours}_{\text{worked}}$, ssn); must delete $\text{Hourly}_{\text{Emps}}$ tuple if referenced Employees tuple is deleted).
- Queries involving all employees easy, those involving just $\text{Hourly}_{\text{Emps}}$ require a join to get some attributes.

4.10.3 Alternative: Just $\textbf{Hourly}_{\textbf{Emps,ContractEmps}}$

- $\text{Hourly}_{\text{Emps}}$: ssn, name, lot, $\text{hourly}_{\text{wages}}$, $\text{hours}_{\text{worked}}$.
- Each employee must be in one of these two subclasses.

4.11 NoSQL

5 Lecture 4

5.1 Relational Query Languages

- Query languages: Allow manipulation and retrieval of data from a database.
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.
- Query Languages != Programming Languages!

- QLs not expected to be "Turing complete".
- QLs not intended to be used for complex calculations.
- QLs support easy, efficient access to large data sets.

5.2 Relational Algebra

More operational, very useful for representing execution plans. Since each operation returns a relation, operations can be composed! (Algebra is 5 basic operations.

5.2.1 Selectiona and projection

Selection (σ) Selects a subset of rows from relation (horizontal).

- Selects rows that satisfy selection condition.
- Output schema of result is same as that of the input relation

E.g. $\sigma_{\text{rating}<9\ \hat{}\ \text{age}>50}(\text{S2})$

Projection (π) Retains only wanted columns from relation (vertical).

- Retains only attributes that are in the projection list.
- Output schema is exactly the fields in the projection list, with the same

names that they had in the input relation. E.g. $\pi_{\text{sname,rating}}(\text{S2})$

- Projection operator has to eliminate duplicates (How do they arise? Why remove them?)
- Relational algebra is set based while SQL is bag (multiset) based

E.g. rows with same age, if project, should only get two ages, however in SQL, we get them all...

5.2.2 Composing multiple operators

Output of one operator can become input to another operator.
E.g. $\pi_{\text{sname,rating}}(\sigma_{\text{rating}>8}(\text{S2}))$

5.2.3 Union, Set Difference & Intersection

5.2.4 Cross-product (X)

Allows us to combine two relations.

5.2.5 Set-difference (−)

Tuples in r1, but not in r2.

5.2.6 Union (∪)

Tuples in r1 and/or in r2. “closed”).

5.3 Relational Calculus

Lets users describe what they want, rather than how to compute it. (Non-procedural, declarative.)