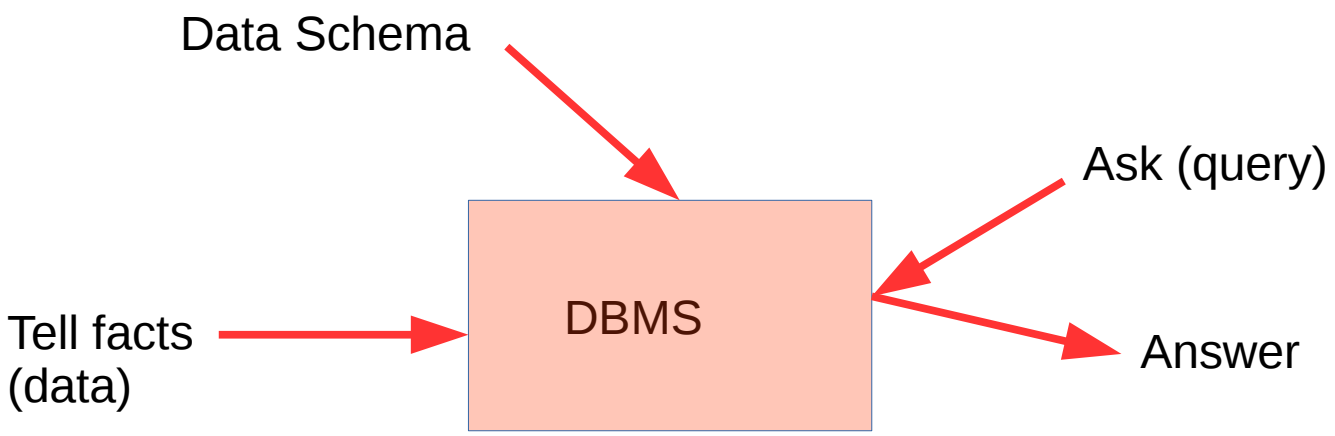
Functional view of Information Management (1)



Need for languages:

- A language to provide the schema (L_{SCHEMA})
- A language to tell facts (L_{TELL})
- A language to ask questions or query language (L_{QUERY})
- A language in which the answer is provided (L_{ANSWER})

In more general terms we will call the DBMS just an Information Manager. Keep in mind that a DBMS is an Information Manager but not every Information Manager is a DBMS.

Functional view of Information Management (3)

The functions will take the following inputs:

SCHEMA(*s*,*M*):

- $s \in L_{SCHEMA}$ is an element of the language used to provide the schema to the IM.
- *M* is an instance of the IM, including the state of the database a the time the SCHEMA operation is executed.

TELL(*t*,*M*):

- $t \in L_{TELL}$ is an element of the language used to tell facts.
- *M* is an instance of the IM, including the state of the database a the time the TELL operation is executed.

QUERY(*q*,*M*):

- $q \in L_{QUERY}$ is an element of the language used to query the IM.
- *M* is an instance of the IM, including the state of the database a the time the QUERY operation is executed.

Datatypes in SQL

- int integer
- real float
- char(n)
- varchar(n) *has terminator*
- date: DATE '2002-09-06'
- time: TIME '18:30:01'
- ENUM('mon','tue', 'wed', ...
- (DATATYPE/DOMAIN/TYPEDEF for *incomparable* types often not supported)

L_{SCHEMA} is an element of the language used to provide the schema to the IM

Creating Relations in SQL

- Creates the **students** relation.
  - » the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.
- Another example: table **enrolledIn** holds information about courses that students take.

```
CREATE TABLE students (
        sid CHAR(10),
        name VARCHAR(20),
        login CHAR(10),
        age INTEGER,
        gpa  FLOAT,
        …);
```

```
CREATE TABLE enrolledIn (
        sid CHAR(20),
        cid CHAR(20),
        grade CHAR(2),
        ...);
```

Primary Key Constraints

- A set of fields is a *key* for a relation if :
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.
  - » *(What if Part 2 false? Called a superkey.)*
  - » *(If there's more than one possible key for a relation, one of them is chosen (by designer) to be the primary key.)*
- E.g., *sid* is a good key for *students*. (What about *name*?) The set {*sid*, *gpa*} is a superkey.

```
CREATE TABLE enrolledIn (
                sid CHAR(20),
                cid CHAR(20),
                grade CHAR(2),
                PRIMARY KEY (sid,cid));
```

(Allows course to be taken only once)

NOTE: SQL is case-insensitive

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO  students
VALUES  (688, 'Smith' , 'smith@ee' , 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., age = 18):

```
DELETE
FROM students S
WHERE S.age = 18
```

L_{QUERY}: The SQL Query Language

"Find all 18 year old students"

- in SQL

```
SELECT *
FROM students S
WHERE S.age=18
```

original table

sid	name	login	age	gpa
666	Jones	jones@cs	18	3.4
688	Smith	smith@eecs	18	3.2
650	Smith	smith@math	20	3.8

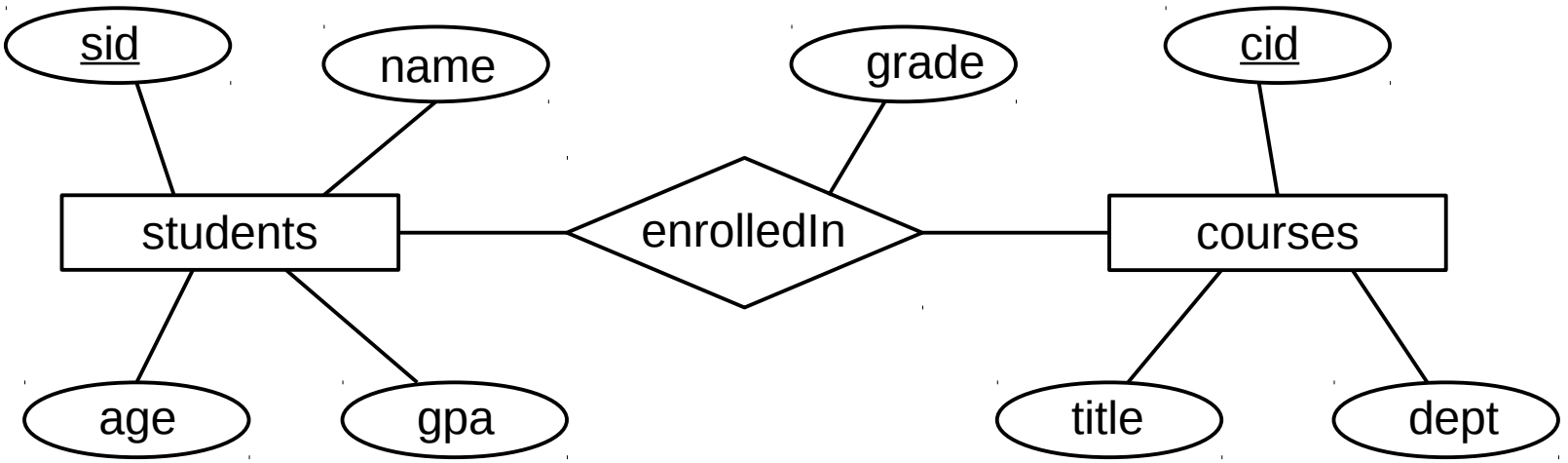
answer

sid	name	login	age	gpa
666	Jones	jones@cs	18	3.4
668	Smith	smith@ee	18	3.2

student(Sid,Name,Gpa,Age)
course(Cid, Title, Dept)
enrolledIn(Sid,Cid,Grade)

L_{ANSWER}: A relation/table instance - *(closure)*

Given the following ER diagram create the tables in SQL



RELATIONAL SCHEMA:

students(sid:int, name:string, gpa:float, age:int, primary key(sid))

courses(cid:int, title:string, dept:string, primary key (cid))

enrolledIn(sid:int, cid:int, grade:string,
primary key (sid, cid),
foreign key (sid) references students,
foreign key (cid) references courses)

SQL –Table Creation

students(sid:int, name:string, age:int, gpa:float, primary key(sid))

students

sid	name	gpa	age
integer	string	float	intger
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	25
95	Bob	3	63

- What is/isn't unique?
  - » what can you tell from looking at a single instance of a table?
  - » You need domain semantics
- What is/isn't a key?
- What is a primary key?

SQL –Table Creation *(3 ways to declare a primary key)*

```
create table students(sid int,
                      name varchar(30),
                      gpa int,
                      age real,
                      primary key (sid));
```

```
create table students(sid int primary key,
                      name varchar(30),
                      gpa int,
                      age real);
```

Only when not a composite key

```
create table students(sid int,
                      name varchar(30),
                      gpa int,
                      age real);

alter table students add constraint primary key (sid);
```

SQL –Table Creation

courses

cid	title	dept
integer	*string*	*string*
101	aaa	chem
102	aaa	cs
103	ccc	math
104	mmm	cs

- What is/isn't unique?
- What is/isn't a key?
- What is a primary key?

Or should there be a composite key (cid,dept) ?

courses

cid	title	dept
integer	*string*	*string*
160:101	aaa	chem
198:102	aaa	cs
640:103	ccc	math
198:104	mmm	cs

```
create table courses(cid int,
                     title varchar(20),
                     dept varchar(10),
                     primary key (cid));
```

SQL – More constraints: FOREIGN KEYS

enrolledIn

sid	cid	grade
integer	integer	char
22	160:101	A
22	198:102	C
22	640:103	B
22	198:104	A
31	198:102	F
31	640:103	B
31	198:104	D
64	160:101	C
64	198:102	A
74	640:103	C

- What is unique?
- What is a key?
- What is a primary key?
  - » What is implied?
- Foreign keys
  - » How does this help?

SQL –Table Creation

Relational Model:

enrolledIn(sid:int, cid:int, grade:string,
primary key (sid, cid),
foreign key (sid) references students,
foreign key (cid) references courses)

```
create table enrolledIn (sid integer,
                         cid integer,
                         grade char(2),
                         primary key(sid,cid),
                        foreign key (sid) references students,
                         foreign key (cid) references courses)
```

Abbreviated form:

```
create table enrolledIn (sid integer references students,
                         cid integer references courses,
                         grade char(2),
                         primary key(sid,cid) )
```

L_{QUERY}: The SQL Query Language

"Find names and logins of all 18 year old students"

- in SQL

```
SELECT  S.name, S.login
FROM  students S
WHERE  S.age=18
```

student(Sid,Name,Gpa,Age)
course(Cid, Title, Dept)
enrolledIn(Sid,Cid,Grade)

original table

sid	name	login	age	gpa
666	Jones	jones@cs	18	3.4
688	Smith	smith@eecs	18	3.2
650	Smith	smith@math	20	3.8

answer

name	login
Jones	jones@cs
Smith	smith@ee

Query answering on a single relation:
a simple procedural specification

```
SELECT  A1, A2,..., An
FROM   R r
WHERE   <condition on r fields>
```

- Consider a variable ranging over all rows of the table in FROM (like a *for*-loop in Java/C++)
- Test the WHERE condition, keeping only rows that are true
- Return for each row, the fields specified in SELECT