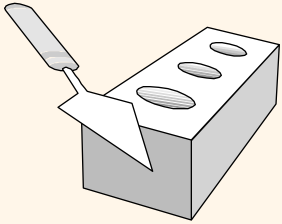
The Relational Model

Chapter 3

Why Study the Relational Model?

- Most widely used model.
  - Vendors: IBM, Informix, Microsoft (Access and SQL Server), Oracle, SAP.
- "Legacy systems" in older models
  - E.G., IBM's IMS
- Competitor: object-oriented model
  - ObjectStore, Versant, and etc.
  - A synthesis emerging: *object-relational model*
    - Informix Universal Server, UniSQL, Oracle, DB2
    - More in scientific computing

Relational Database: Definitions

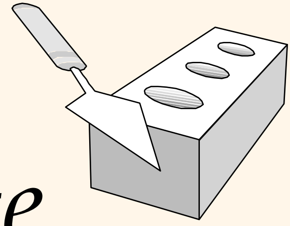


- *Relational database:* a set of *relations*
- *Relation:* made up of 2 parts:
  - *Relation Schema* : specifies name of relation, plus name and type of each column.
    - E.g., Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).
  - *Relation Instance* : a *table,* with rows and columns. #Rows = *cardinality*, #fields = *degree / arity.*
- Can think of a relation as a *set* of rows or *tuples* or *records* (i.e., all rows are distinct).

Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5, all rows distinct



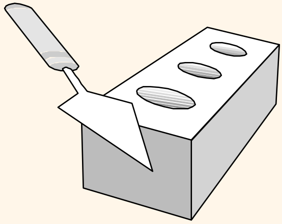
SQL: Structured Query Language

- Developed by IBM (system R) in the 1970s
- The most widely used language for creating, manipulating, and querying relational DBMS.
- Need for a standard since it is used by many vendors
- Standards:
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision)
  - SQL-99 (major extensions)
  - SQL-03, SQL-06 , SQL-08, SQL-11, SQL-16, SQL-19, SQL-23

SQL: Structured Query Language

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

Simple SQL Queries



sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- To find the names and login of all 18 year old students:

```
SELECT  name, login
FROM  Students
WHERE  age=18
```

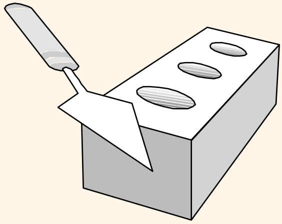
name	login
Jones	jones@cs
Smith	smith@eecs

- To find all student data, replace the first line:

```
SELECT  *
FROM  Students
WHERE  age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2

Querying Multiple Relations



Given the following instances of Enrolled and Students, list all the students who get grade 'A' in any course along with the course id:

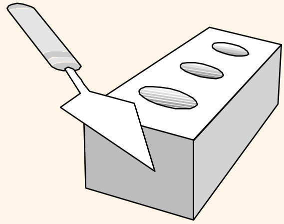
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Answer:

S.name	E.cid
Smith	Topology112

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade='A'
```



Creating Relations in SQL

```
CREATE TABLE Students (sid: CHAR(20),
                       name: CHAR(20),
                       login: CHAR(10),
                       age: INTEGER,
                       gpa: REAL)
```

- Creates the Students relation.
  - The type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

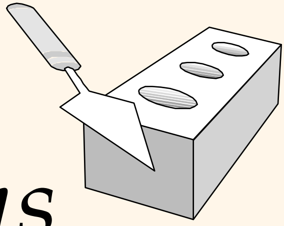
sid	name	login	age	gpa

Creating Relations in SQL

```
CREATE TABLE Enrolled  (sid: CHAR(20),
                        cid: CHAR(20),
                        grade: CHAR(2))
```

- The Enrolled table holds information about courses that students take.

sid	cid	grade



Destroying and Altering Relations

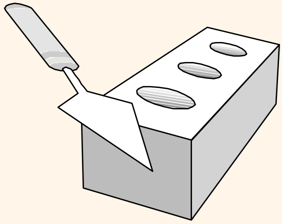
DROP TABLE Students

- Destroys the relation Students. The schema information *and* the tuples are deleted.

ALTER TABLE Students
ADD COLUMN firstYear integer

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

Adding and Deleting Tuples



- Can insert a single tuple using:

```
INSERT
INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@umn', 18, 3.2)
```

☛ *The list of columns is optional*

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

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

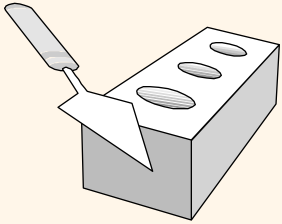
Updating Tuples

- Update a single tuple

```
UPDATE Students S
SET S.age = S.age +1, S.gpa = S.gpa -1
WHERE S.sid = ‘54832’
```

- Update multiple tuples

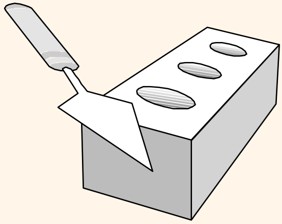
```
UPDATE Students S
SET S.age = S.age +1, S.gpa = S.gpa -1
WHERE S.gpa > 3.3
```



Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database; e.g., *domain constraints.*
  - 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.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!

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.
     - Part 2 false? A *superkey*.
     - If there are more than one key for a relation, one of the keys is chosen (by DBA) to be the *primary key* while other keys are considered *candidate key*
- E.g., *sid* is a key for Students. (What about *name*?) The set {*sid, gpa*} is a superkey.

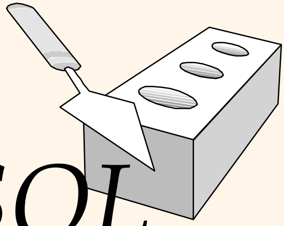
Primary and Candidate Keys in SQL

- "For a given student and course, there is a single grade."

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

- "Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade."

```
CREATE TABLE Enrolled (sid CHAR(20)
                       cid  CHAR(20),
                       grade CHAR(2),
                       PRIMARY KEY  (sid),
                       UNIQUE (cid, grade) )
```

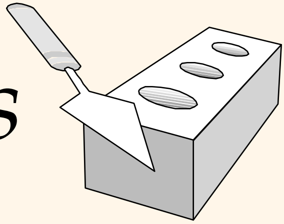


Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key* (*specified using* PRIMARY KEY) .

- Used carelessly, an IC can prevent the storage of database instances that arise in practice!

Enforcing Primary Key Constrains



Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Having this table: The following transactions are ***REJECTED:***
  - INSERT INTO Students (sid, name, login, age, gpa)
    VALUES ('53688', 'Mike', 'Mike@cs', 17, 3.4)
  - INSERT INTO Students (sid, name, login, age, gpa)
    VALUES (*null*, 'Mike', 'Mike@cs', 17, 3.4)
  - INSERT INTO Students (sid, name, login, age, gpa)
    VALUES ('53784', 'Mike', 'Mike@cs', 'twenty', 3.4)
  - UDPATE Students S SET S.sid = '53688'
    WHERE S.sid = ('53666')
- There are no problems for *Deletion*