

RELATIONAL MODEL



Relational Data Model

- Introduced by Ted Codd (early 70') (Turing Award, '81)

- Relational data model contributes:
 1. Separation of logical and physical data models (data independence)
 2. Query languages
 3. Query optimization (key to commercial success)

Relations

account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-202	450
Brookline	A312	600

- Rows (tuples, records)
- Columns (attributes)
- Tables (relations)

- Why relations?

Relations

□ Mathematical relations (from set theory):

Given 2 sets $R = \{1, 2, 3, 5\}$, $S = \{3, 4\}$

- $R \times S = \{(1,3), (1,4), (2,3), (2,4), (3,3), (3,4), (5,3), (5,4)\}$
- A relation between R and S is any subset of $R \times S$
e.g., $\{(1,3), (2,4), (5,3)\}$

□ Database relations:

Given attribute domains:

$\text{bname} = \{\text{Downtown}, \text{Brighton}, \dots\}$

$\text{acct_no} = \{A-101, A-102, A-203, \dots\}$

$\text{balance} = \{\dots, 400, 500, \dots\}$

$\{(\text{Downtown}, A-101, 500),$
 $(\text{Brighton}, A-202, 450),$
 $(\text{Brookline}, A-312, 600)\}$

$\text{account} \text{ subset of } \text{bname} \times \text{acct_no} \times \text{balance}$

Storing Data in a Table

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

- Data about individual students
- One row per student
- How to represent course enrollment?

Storing More Data in Tables

- Students may enroll in more than one course
- Most efficient: keep enrollment in separate table

Enrolled

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

Linking Data from Multiple Tables

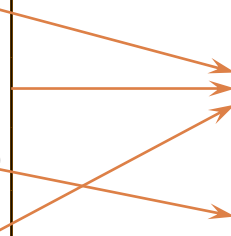
- How to connect student data to enrollment?
- Need a *Key*

Enrolled

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

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



Relational Data Model: Formal Definitions

- *Relational database*: a set of *relations*.
- *Relation*: made up of 2 parts:
 - *Instance* : a *table*, with rows and columns.
 - *#rows = cardinality*
 - *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)
 - *#fields = degree / arity*
- Can think of a relation as a *set* of rows or *tuples*.
 - i.e., all rows are distinct

In other words...

- Data Model – a way to organize information
- Schema – one particular organization,
 - ▣ i.e., a set of fields/columns, each of a given type
- Relation
 - ▣ a name
 - ▣ a schema
 - ▣ a set of tuples/rows, each following organization specified in schema

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, arity (degree) = 5 , all rows distinct

SQL - A language for Relational DBs

- SQL: standard language (based on SEQUEL in System R (IBM now DB2))
- Data Definition Language (DDL)
 - create, modify, delete relations
 - specify constraints
 - administer users, security, etc.
- Data Manipulation Language (DML)
 - Specify *queries* to find tuples that satisfy criteria
 - add, modify, remove tuples

SQL Overview

- ❑ CREATE TABLE <name> (<field> <domain>, ...)
- ❑ INSERT INTO <name> (<field names>)
VALUES (<field values>)
- ❑ DELETE FROM <name>
WHERE <condition>
- ❑ UPDATE <name>
SET <field name> = <value>
WHERE <condition>
- ❑ SELECT <fields>
FROM <name>
WHERE <condition>

Creating Relations in SQL

- Creates the Students relation.

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

- Note: the type (**domain**) of each field is specified, and enforced by the DBMS

- whenever tuples are added or modified.

- Another example: the Enrolled table holds information about courses students take.

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

Adding and Deleting Tuples

- Can insert a single tuple using:

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

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

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

- ☞ **Powerful variants of these commands are available; more later!**

Updating Tuples

- Can update all tuples, or just those matching some criterion

```
UPDATE Students  
SET sid = sid + 100000
```

```
UPDATE Enrolled  
SET grade = 'E'  
WHERE grade = 'F'
```

Integrity Constraints over Relations

□ Keys

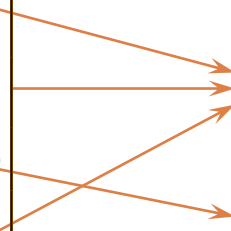
- ▣ Keys are a way to associate tuples in different relations
- ▣ Keys are one form of integrity constraint (IC)

Enrolled

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

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



Primary Keys - Definitions

- A set of fields is a superkey if:
 - ▣ No two distinct tuples can have same values in all key fields
- A set of fields is a candidate key for a relation if :
 - ▣ It is a superkey
 - ▣ No subset of the fields is a superkey
- >1 candidate keys for a relation?
 - ▣ one of the keys is chosen (by DBA) to be the *primary key*.
- E.g.
 - ▣ *sid* is a key for Students.
 - ▣ What about *name*?
 - ▣ The set {*sid*, *gpa*} is a superkey.

Primary Key Constraints

A set of fields is a key for a relation if :

- ▣ No two distinct tuples can have same values in all key fields, and
- ▣ This is not true for any subset of the key.
 - Part 2 false? A superkey.
 - If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the primary 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

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

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

VS.

“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.”

- Used carelessly, an IC can prevent storage of database instances that should be permitted!

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

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

Foreign Keys

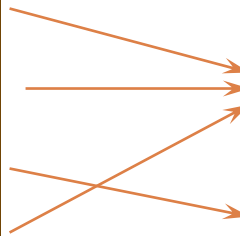
- A Foreign Key is a field whose values are keys in another relation.

Enrolled

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

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



Foreign Keys, Referential Integrity

- **Foreign key : Set of fields in one relation used to 'refer' to tuples in another relation.**
 - ▣ Must correspond to primary key of the second relation.
 - ▣ Like a 'logical pointer'.
- **E.g. *sid* in *Enrolled* is a foreign key referring to *Students*:**
 - ▣ Enrolled(*sid*: string, *cid*: string, *grade*: string)
 - ▣ If all foreign key constraints are enforced, referential integrity is achieved (i.e., no dangling references.)

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses.

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

Enrolled

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

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

Enforcing Referential Integrity

- Consider Students and Enrolled; sid in Enrolled is a foreign key that references Students.
 - ▣ What should be done if an Enrolled tuple with a non-existent student id is inserted? (Reject it!)
 - ▣ What should be done 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.
 - (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value null, denoting 'unknown' or 'inapplicable'.)
- Similar if primary key of Students tuple is updated

Referential Integrity in SQL

- Default is NO ACTION (delete/update is rejected)
- CASCADE (also delete all tuples that refer to deleted tuple)
- SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET DEFAULT )
```


Other 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!

- ICs are based upon the semantics of the realworld enterprise that is being described in the database relations.
- We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
 - ▣ An IC is a statement about all possible instances!
- Key and foreign key ICs are the most common