Review of Lecture 1-1

- Course logistics
- Motivation to DB
  - Redundancy and inconsistency, concurrent access
- Database and DBMS
- Schema and instance
  - 3 level of abstraction
  - Data independence
- DDL and DML
  - SQL: both and control
- Limitation of DB

Lecture 2-1: Today's Plan

- Relational data model
  - Set, simple attribute
  - Order of row vs. order of column
- Integrity constraints in relational data model
  - Domain
  - Primary Key
  - Foreign Key

RELATIONAL DATA MODEL

CHAPTER 02

Collin Roberts
University of Waterloo

Chapter Outline

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations

Suggested Readings

- Textbook Chapter 05

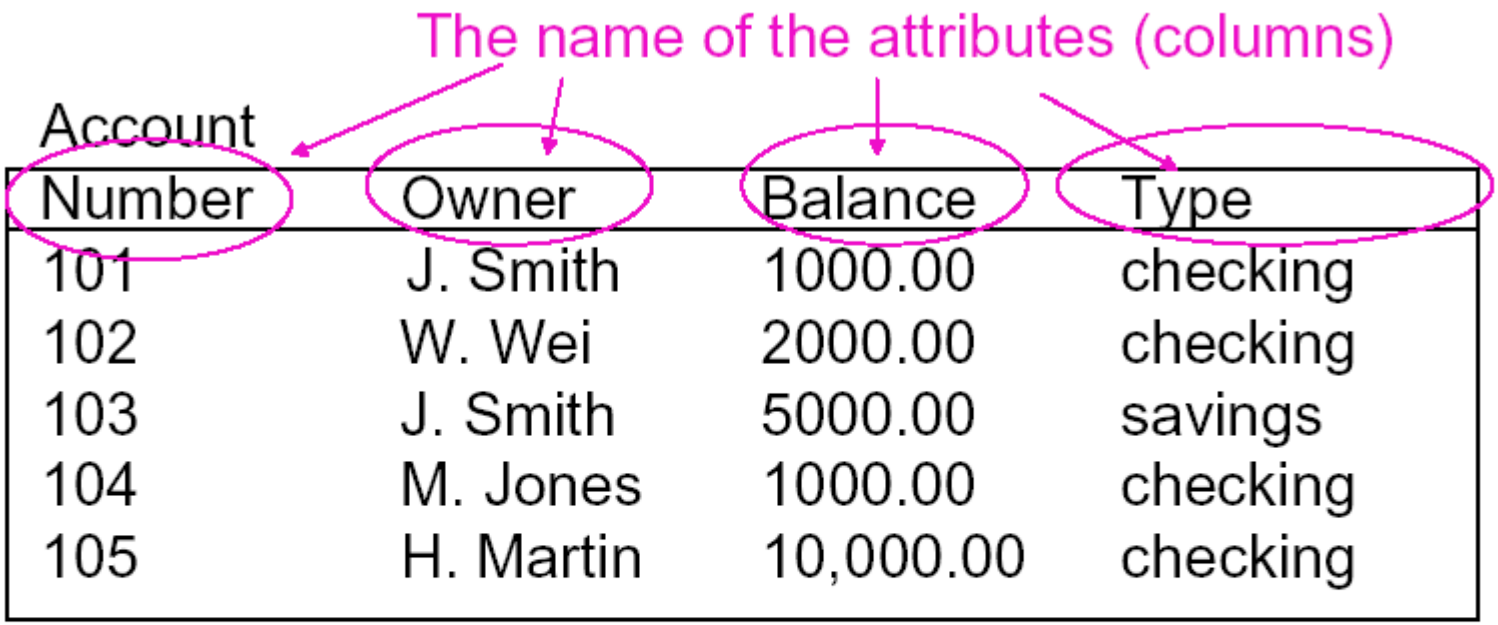
Why Relational Data Model?

- Most widely used model.
  - Vendors: IBM, Informix, Microsoft, Oracle, Sybase, etc.
- Recent competitors: object-oriented model, semi-structured (XML) model
  - ObjectStore, Versant, Tamino
  - A synthesis emerging:
    - *object-relational model:* Informix Universal Server, UniSQL, O2, Oracle, DB2
    - XML-enabled databases: Oracle, DB2, SQLServer

Relation – Informal Definition

- RELATION: A table of values
  - A relation may be thought of as a set of rows with several columns.
  - Each row represents a real-world entity or relationship.
  - Each column typically is called by its column name or column header
    - Called attribute
  - The values for each attribute must come from a domain (all possible values)
  - Each row has a set of values that uniquely identifies that row in the table.
    - Called primary key or simply key
    - Sometimes row-ids or sequential numbers are (automatically) assigned to identify the rows in the table.

Example: A Relation



Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Attribute domains:

Number must be a 3-digit number

Owner must be a 30-character string

Type must be “checking” or “savings”

Example: Relational Schema

The schema for the relation

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The schema sets the structure of the relation. You can think of the schema as the *definition* of the relation. (Note, the schema specifies more information than what is shown.)

Example: Instance

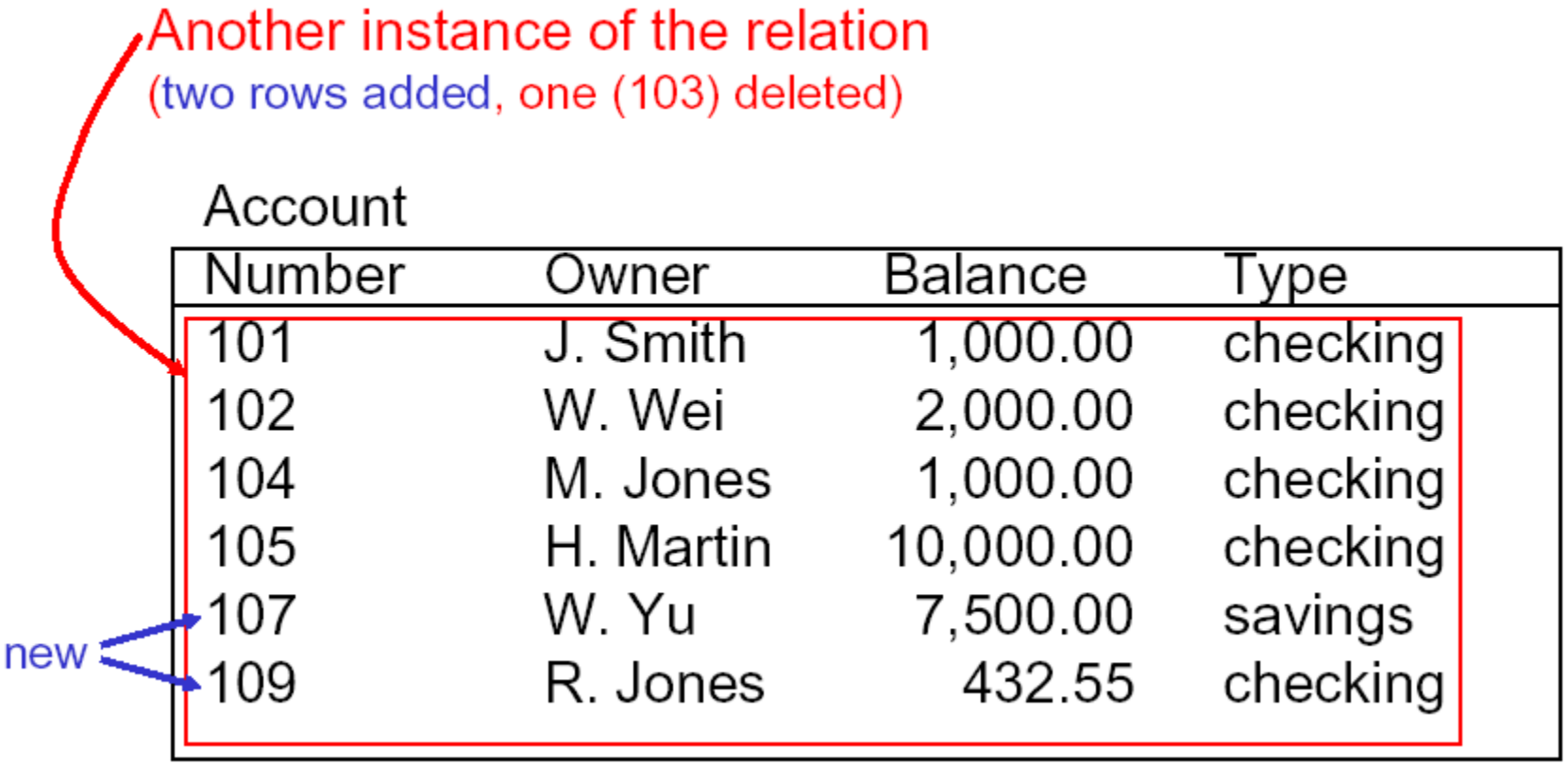
An instance of the relation...

the current contents or data in the relation.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Example: Another Instance



Account

Number	Owner	Balance	Type
101	J. Smith	1,000.00	checking
102	W. Wei	2,000.00	checking
104	M. Jones	1,000.00	checking
105	H. Martin	10,000.00	checking
107	W. Yu	7,500.00	savings
109	R. Jones	432.55	checking

Important Note

- We can only run a query over an instance to check whether the query is wrong, but not whether it is correct.
  - **If the result is wrong, the query must be wrong**
  - **If the result is correct, the query might be correct, but might be wrong as well**
- A correct query MUST return the right answers in all instances

Alternative Terminology (FYI)

The intension of the table

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The extension of the table

Row/Tuple

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Each entry in the relation is called a **row** or a **tuple**.
Sometimes an entry in the relation is called a record.
Order of tuples is irrelevant. Why?

Degree and Cardinality

Degree or arity of a relation is the number of attributes

Degree of this relation (or table) is 4 because there are 4 attributes

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Cardinality of this instance is 5 (because there are 5 rows)

Cardinality of a relation = the number of rows in the current instance

Terminology Summary

Informal Terms	Formal Terms
Table	Relation
Column Header/Field	Attribute
Row/Record	Tuple
Possible Values in a column	Domain
Table Definition	Schema of a Relation/Intension
Table Content	Instance/Extension

Relation – Formal Definition

- Given sets D_1, D_2, …. D_n a **relation** *r* is a subset of D_1 x D_2 x … x D_n
  Thus a relation is a set of n-tuples
  $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
- Each attribute of a relation has a name
- Set of allowed values for attribute is called the **domain** of the attribute
- Attribute values are required to be **atomic**, that is, indivisible
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic
- The special value *null* is a member of every domain

Relational Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

  E.g.: *account* : information about accounts
  deposit: information about deposits into accounts
  check : information about checks
- Storing all information as a single relation such as *bank*(*account-number, balance, customer-name*, *deposit-date, deposit-amount*..) results in

  •repetition of information (e.g. two customers own an account)
  •the need for null values (e.g. represent a customer without an account)

Example: Relational Database

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00

Check	Account	Check-number	Date	Amount
	101	924	10/23/00	125.00
	101	925	10/24/00	23.98

Integrity Constraints (IC)

- 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 in Relational Database

- Domain constraints
- Primary key constraints
- Foreign key constraints

Domain constraints

- Value for an attribute must be atomic
  - For example, phone number
- Value for an attribute must come from the domain of the attribute
  - For example, int for int, float for float, string for string etc.

Primary Key Constraints

- Primary key constraints: each relation must have a primary key
  - Value for primary key must be unique and non-null (not empty)
- Review question: why PK?

Primary Keys

- Primary key – minimum set of attributes that is a unique identifier for a tuple/row
  - sid is primary key for Students
  - cid is primary key for Courses
  - A table might have several possible PKs. Each is called a candidate key and one of them is designated as PK.
  - PK is often underlined

- If a table has multiple possible PKs, each is called a candidate keys and one will be designated as PK

Example: Primary Keys

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00

Check	Account	Check-number	Date	Amount
	101	924	10/23/98	125.00
	101	925	10/24/98	23.98

Each Relation has a key…. where the values must be unique.

Exercise: Candidate Key?

- Student(SID, E-mail, Address, Phone, Birthday)
  - SID?
  - E-mail?
  - Phone?
  - (SID, Address)

- F20Courses (CID, Title, Time, Location, Cap)

Why Minimal in PK?

- PK: “Minimal” set of attributes whose values are unique for each row
  - Its value must be unique and not empty
- Why minimal?
  - SID is the PK in the Student table
  - Why can’t we make (SID, Name) as the PK?

Discussion: What is Wrong?

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00
	106	5	12/5/00	555.00

Is this legal? If not, how do we prevent it from happening?

Foreign Key and Referential Integrity

- Foreign Key: set of fields in one relation that is used to refer to a tuple in another relation
  - Often (but NOT always) corresponds to the primary key of the second relation like a "logical pointer".
  - The second relation could be the same relation, that is, pointing to itself (Ex. the supervisor FK in the employee table).
- E.g. *sid* 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.
  - Can you name a data model w/o referential integrity?
    - Links in HTML!

Definitions of FK Constraints

- Referencing relation (R1): points to (refers to) other relation
  - For example, the Enrolled relation
- Referenced relation (R2): pointed to (referred to) by other relation
  - For example, the Students relation
- If FK exists, the value in the foreign key column (or columns) of the referencing relation R1 must be either:
  - a unique value in the referenced relation R2, or.
  - null.
- Note: PK is mandatory but FK is optional

Example: Foreign Key

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00
	~~106~~	~~5~~	~~12/5/00~~	~~555.00~~
	NULL	6	9/10/20	100

We say that Deposit.Account is a foreign key that references Account.Number. If the DBMS enforces this constraint we say we have **referential integrity**.

Exercise: Foreign Key

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00
	~~106~~	~~5~~	~~12/5/00~~	~~555.00~~
	NULL	6	9/10/20	100

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.

Example: Enforcing FK

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

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

What if Jones is deleted from Students table?

Example: Enforcing FK

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

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

What if Jones's SID is changed to 111111?

Where Does IC Come From?

- ICs are based upon the semantics of the real-world 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!
  - From example, we know *name* is not a key, but the assertion that *sid* is a key is given to us.
- Key and foreign key ICs are the most common; more general ICs supported too.

Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the data model, but at the application level (you need to write code to describe it)
    - E.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
    - Or, “manager salary must be higher than regular worker salary”
    - Or, “A student with average below 80 cannot take more than 5 courses”
  - A constraint specification language may have to be used to express these
    - SQL-99 allows triggers and ASSERTIONS to allow for some of these

Specifying a Relation Schema

- Select the relations, with a name for each table.
- Select attributes for each relation and give the domain for each attribute.
- Specify the key(s) for each relation.
- Specify all appropriate foreign keys.

Exercise - FK

Teacher (Number, Name, Office, E-mail)

Course (Number, Name, Description)

Taught-By (Quarter, Course, Section, Teacher, TimeDays)

Student (Number, Name, Major, Advisor)

Completed (Student, Course, Quarter, Section, Grade)

Exercise – FK (Answer)

Teacher (Number, Name, Office, E-mail)

Course (Number, Name, Description)

Taught-By (Quarter, Course, Section, Teacher, TimeDays)

Student (Number, Name, Major, Advisor)

Completed (Student, Course, Quarter, Section, Grade)

Foreign keys in the Completed table are shown above.

Exercise – PK and FK

(Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SID, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SID, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

Identify the primary keys and foreign keys for this schema.

Summary of Relational Model I

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

Summary of Relational Model II

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - In addition, we *always* have domain constraints.
- Powerful and natural query languages exist.
- Rules to translate ER to relational model

Summary of 2-1

- Relational data model
- Integrity constraints in relational data model