# CS 325 Week 3-2

Intro to the relational model (DB Reading Packet 3)

# Intro to the RELATIONAL model

- Recall: Developed by E.F. Codd (1970 at IBM)
  - Based on relational algebra, a branch of mathematics.
- Initial Challenge
  - Seen as a theoretical breakthrough but considered impractical due to the high computer overhead required.
  - Early 1970s hardware, software, and memory costs made it seem too slow and inefficient.

# Intro to the RELATIONAL model

- Recall: Developed by E.F. Codd (1970 at IBM)
  - Based on relational algebra, a branch of mathematics.
- Initial Challenge
  - Seen as a theoretical breakthrough but considered impractical due to the high computer overhead required.
  - Early 1970s hardware, software, and memory costs made it seem too slow and inefficient.
- Technological Advances:
  - As hardware and operating systems improved and became more affordable, the overhead became manageable.

# Intro to the RELATIONAL model

- The DBMS supporting a relational mode is creating an abstraction that your data is in the form of relational tables (relations, or tables that meet the relational criteria)

# Intro to the RELATIONAL model

- The DBMS supporting a relational mode is creating an abstraction that your data is in the form of relational tables (relations, or tables that meet the relational criteria)
- DBMS Independence
  - The relational model allows database designs to be created independently of the DBMS.
  - Similar to how algorithms can be discussed without referencing specific programming languages.

# Intro to the RELATIONAL model

- The DBMS supporting a relational mode is creating an abstraction that your data is in the form of relational tables (relations, or tables that meet the relational criteria)
- DBMS Independence
  - The relational model allows database designs to be created independently of the DBMS.
  - Similar to how algorithms can be discussed without referencing specific programming languages.
- Why This Matters
  - You can design a database (determine the relational tables) before deciding which DBMS to use.This flexibility makes it easier to adapt your design to different DBMS platforms.

# Intro to the RELATIONAL model

- The DBMS supporting a relational mode is creating an abstraction that your data is in the form of relational tables (relations, or tables that meet the relational criteria)
  - powerful and elegant!
- Relational model allows for the ability to create DBMS-independent database designs
  - this includes, for a relational database, what tables? what columns in those tables? how are they related?

# Intro to the RELATIONAL model

- A relational databases is a collection of relations
  - and, what is a relation?

# Intro to the RELATIONAL model

- A relational databases is a collection of relations
  - and, what is a relation?
    - being formal:
    - Ulmann, 2nd ed., p. 19:
    - a subset of the Cartesian product of a list of domains

# Question:

Then, what do you think is a **relation schema** in the context of a relational database?

1) A set of constraints that enforce data integrity in a table
2) A list of possible values that an attribute can have, including null values
3) A predefined query structure used to retrieve data from multiple tables
4) A finite sequence of unique attribute names that define the structure of a table

# Intro to the RELATIONAL model

- A relational databases is a collection of relations
  - and, what is a relation?
    - being formal:
    - Ulmann, 2nd ed., p. 19:
    - a subset of the Cartesian product of a list of domains
    - (pulling from Sunderraman here...)
    - relation scheme/schema:
      - (in math terms) a finite sequence of unique attribute names
      - (attribute is kind of like the math equivalent of a column)

# Intro to the RELATIONAL model

- (pulling from Sunderraman here...)
- relation scheme/schema:
  - (in math terms) a finite sequence of unique attribute names
  - (attribute is kind of like the math equivalent of a column)
  - You can give this relation scheme a name -- so, for example:

```
employees = (empl_id, empl_last_name, empl_str_addr, empl_salary)
```

# Intro to the RELATIONAL model

- (pulling from Sunderraman here...)
- relation scheme/schema:
  - (in math terms) a finite sequence of unique attribute names
  - (attribute is kind of like the math equivalent of a column)
- an attribute name A is associated with a domain, dom(A), a set of values, which includes the special value null

# Intro to the RELATIONAL model

- relation scheme/schema:
  - (in math terms) a finite sequence of unique attribute names
  - (attribute is kind of like the math equivalent of a column)
- an attribute name A is associated with a domain, dom(A), a set of values, which includes the special value null
  - ex: dom(empl_id) could be the set of integers between 1000 and 9999 and the special value null

```
empl_id NUMBER,
CHECK (empl_id BETWEEN 1000 AND 9999 OR empl_id IS NULL)
```

# Intro to the RELATIONAL model

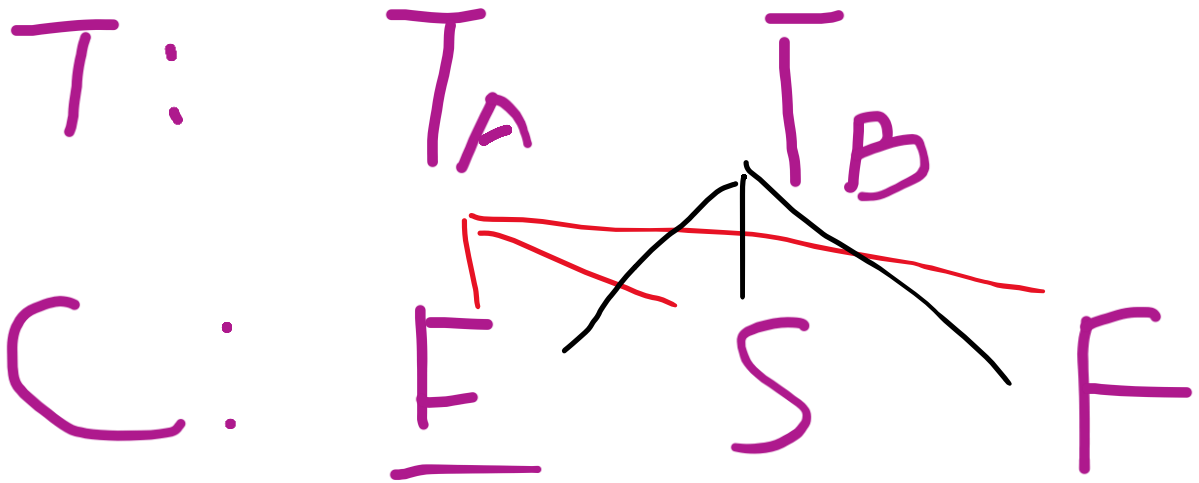
- relation scheme/schema:
  - (in math terms) a finite sequence of unique attribute names
  - (attribute is kind of like the math equivalent of a column)
- an attribute name A is associated with a domain, dom(A), a set of values, which includes the special value null
- given a relation scheme R = (A1, A2, ... An), a relation r on the scheme R is defined as any finite subset of the Cartesian product
  - dom(A1) x dom(A2) x ... x dom(An)

# Refresh: Refresh: Cartesian product

$$\text{Set } T \longrightarrow \text{ a set of Teachers } \longrightarrow \text{ Set } T = \{T_A, T_B\}$$
$$\text{Set } C \longrightarrow \text{ a set of Courses } \longrightarrow \text{ Set } C = \{ \text{ English, Spanish, French } \}$$

# Refresh: Refresh: Cartesian product

$$\text{Set } T \longrightarrow \text{ a set of Teachers } \longrightarrow \text{ Set } T = \{T_A, T_B\}$$
$$\text{Set } C \longrightarrow \text{ a set of Courses } \longrightarrow \text{ Set } C = \{ \text{ English, Spanish, French } \}$$



# Refresh: Refresh: Cartesian product

$$\text{Set } T \longrightarrow \text{ a set of Teachers } \longrightarrow \text{ Set } T = \{T_A, T_B\}$$
$$\text{Set } C \longrightarrow \text{ a set of Courses } \longrightarrow \text{ Set } C = \{ \text{ English, Spanish, French } \}$$

$$\{(T_{A'} \text{ English}), (T_{A'} \text{ Spanish}), (T_{A'}, \text{ French})$$
$$(\mathrm{T}_{\mathrm{B}'} \text{ English}), (\mathrm{B}_{\mathrm{B}'} \text{ Spanish}), (\mathrm{T}_{\mathrm{B}'}, \text{ French})\}$$

$T \times C$

# Intro to the RELATIONAL model

- given a relation scheme R = (A1, A2, ... An), a relation r on the scheme R is defined as any finite subset of the Cartesian product
  - dom(A1) x dom(A2) x ... x dom(An)
  - reminder: Cartesian product of sets A and B: A x B
  - the set of all ordered pairs of ONE element from A and ONE element from B

# Question

Given the following two sets:

Set A = {1, 3, 5}

Set B = {a, b}

What is the **Cartesian product** of Set A and Set B (A x B)?

1) {(1, a), (1, b), (3, a), (5, b)}
2) {(1, a), (1, b), (3, a), (3, b), (5, a), (5, b)}
3) {(a, 1), (3, a), (a, 5), (1, b), (b, 3), (5, b)}
4) {(1, a), (1, b), (3, a), (5, a), (5, b)}

# Intro to the RELATIONAL model

- given a relation scheme R = (A1, A2, ... An), a relation r on the scheme R is defined as any finite subset of the Cartesian product
  - dom(A1) x dom(A2) x ... x dom(An)
  - reminder: Cartesian product of sets A and B: A x B
  - the set of all ordered pairs of ONE element from A and ONE element from B
- happily, a relation is a FINITE subset of this Cartesian product!
  - and note that ONE element of this relation is the ordered set of ONE value of A1, ONE value of A2, ... ONE value of An
  - ...sounds kind of like a ROW, doesn't it?
  - ...in Math, that's also called a tuple

# Intro to the RELATIONAL model

- happily, a relation is a FINITE subset of this Cartesian product!
  - and note that ONE element of this relation is the ordered set of ONE value of A1, ONE value of A2, ... ONE value of An
  - ...sounds kind of like a ROW, doesn't it?
  - ...in Math, that's also called a tuple
- (And in practice, we'll tend to pick subsets of this Cartesian product that we find "useful" or have some sort of meaning)

# Intro to the RELATIONAL model

- (And in practice, we'll tend to pick subsets of this Cartesian product that we find "useful" or have some sort of meaning)
- SO, for relation scheme
  - employees = (empl_id, empl_last_name, empl_str_addr, empl_salary)
- a relation might be:
  - {(1111, 'Jones', '111 Ash Str', 20000),  <-- 1 element of this set, a tuple!
  - (2222, ‘Alice', '123 Elm Str', 25000)}  <-- another element of this set!

# Intro to the RELATIONAL model

- a relational database scheme D is a finite set of relation schemes {R1, R2, ... Rm}

# Intro to the RELATIONAL model

- a relational database scheme D is a finite set of relation schemes {R1, R2, ... Rm}
- and a relational database on a relational database scheme D is a set of relations {r1, r2, ... rm} where each ri is a relation on the corresponding relation scheme Ri

# Intro to the RELATIONAL model

- a relational database scheme D is a finite set of relation schemes {R1, R2, ... Rm}
- and a relational database on a relational database scheme D is a set of relations {r1, r2, ... rm} where each ri is a relation on the corresponding relation scheme Ri
- if you think of a relation as a tabular thing, a tuple in a relation is like a row in that tabular thing, and an attribute of the relation is like a column in that tabular thing

# Intro to the RELATIONAL model

- if you think of a relation as a tabular thing, a tuple in a relation is like a row in that tabular thing, and an attribute of the relation is like a column in that tabular thing
- We'd like a tuple/row to hold data that pertains to some "thing" or portion of a "thing"

# Question:

Then, what you do think is a cell in the context of a relational database table?

1) It defines the relationship between two tables or rows
2) The point/intersection where a row and a column meet, representing a single attribute's value within a specific tuple
3) The entire collection of data within a row, encompassing all columns
4) A placeholder for missing values in a table that has not been filled yet

# Intro to the RELATIONAL model

- if you think of a relation as a tabular thing, a tuple in a relation is like a row in that tabular thing, and an attribute of the relation is like a column in that tabular thing
- We'd like a tuple/row to hold data that pertains to some "thing" or portion of a "thing“
- a cell of a relation/table is the intersection of a row and a column, or a single attribute's value within a particular tuple
  - ‘'Jones’ is a cell of the tuple {(1111, 'Jones', '111 Ash Str', 20000)

# Intro to the RELATIONAL model

- NOW -- not everything a person calls a "table" MEETS the relation definition
- (but we are going to try to make sure the tables we create for a relational database DO meet the definition!)

# Intro to the RELATIONAL model

- NOW -- not everything a person calls a "table" MEETS the relation definition
- (but we are going to try to make sure the tables we create for a relational database DO meet the definition!)
- e.g., if you follow the before-mentioned definitions, you NEVER have a multi-valued cell! EACH cell contains either NO value (the special value null) or ONE value (it has ONE from the domain of that attribute, since null might BE in that domain!)

# Intro to the RELATIONAL model

- if you recall your basic rules of sets, a thing is either a member of a set or it is not – there is NO concept of being in a set "multiple times“.

# Intro to the RELATIONAL model

- if you recall your basic rules of sets, a thing is either a member of a set or it is not – there is NO concept of being in a set "multiple times“.
- THAT's why a relation (or a table meeting its rules) cannot have duplicate ROWS (cannot have duplicate TUPLES)

# Intro to the RELATIONAL model

- if you recall your basic rules of sets, a thing is either a member of a set or it is not – there is NO concept of being in a set "multiple times“.
- THAT's why a relation (or a table meeting its rules) cannot have duplicate ROWS (cannot have duplicate TUPLES)
- if you recall your basic rules of sets, we don't really care about the order of the elements in a set -- or, the order of the elements is not SIGNIFICANT
- ...so, the order of the rows or tuples in a relation is not supposed to be significant.

# Intro to the RELATIONAL model

- each attribute within a relation must have a unique name (that's straight from the relation scheme definition we mentioned before
  - relation scheme/schema: (in math terms) a finite sequence of unique attribute names

# Intro to the RELATIONAL model

- each attribute within a relation must have a unique name (that's straight from the relation scheme definition we mentioned before
- all of the values for a particular attribute within a relation/table must be from that attribute's domain

# Question

Consider the relation structure:

dept(dept_name, dept_num, dept_loc)

How many columns/attributes/fields are in the relation dept, according to its relation structure?

1) 4
2) 3
3) 2
4) cannot tell from relation structure form

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- functional dependency is a relationship between or among attributes,

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- functional dependency is a relationship between or among attributes,
- SUCH THAT, given the value of one attribute, A, you can look up/obtain the value of another attribute, B
  - if you can do that, then we say that B is functionally dependent on A

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- functional dependency is a relationship between or among attributes,
- SUCH THAT, given the value of one attribute, A, you can look up/obtain the value of another attribute, B
  - if you can do that, then we say that B is functionally dependent on A
  - ...that A determines B
- CONSIDER the relation:
  - advisor = (ADVISOR_ID, advisor_lname, advisor_fname, advisor_phone)

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- CONSIDER the relation:
  - advisor = (ADVISOR_ID, advisor_lname, advisor_fname, advisor_phone)
  - IF, say, given the value of advisor_id, I can look up/obtain the value of advisor_phone, then I'd say that advisor_phone is FUNCTIONALLY DEPENDENT on advisor_id.

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- CONSIDER the relation:
  - advisor = (ADVISOR_ID, advisor_lname, advisor_fname, advisor_phone)
  - IF, say, given the value of advisor_id, I can look up/obtain the value of advisor_phone, then I'd say that advisor_phone is FUNCTIONALLY DEPENDENT on advisor_id.
  - We also may say that advisor_id DETERMINES advisor_phone

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- I've seen it argued (from another textbook) that the storage and retrieval of functional dependencies is arguably the only reason for having a database?!

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- I've seen it argued (from another textbook) that the storage and retrieval of functional dependencies is arguably the only reason for having a database?!
- a little notation:
  - If I say that B is functionally dependent on A (or that A determines B) then that can be written:
  - A -> B

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- a little notation:
  - If I say that B is functionally dependent on A (or that A determines B) then that can be written:
  - A -> B
- So, I could write:
  - advisor_id -> advisor_phone
  - So how should we read it?

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- a little notation:
  - If I say that B is functionally dependent on A (or that A determines B) then that can be written:
  - A -> B
- So, I could write:
  - advisor_id -> advisor_phone
- and, when you use the notation
  - A -> B
  - ...we say that A is the DETERMINANT in that functional dependency

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- and, when you use the notation
  - A -> B
  - ...we say that A is the DETERMINANT in that functional dependency
- So, in
  - advisor_id -> advisor_phone
  - So, ...we say what? Which is the DETERMINANT?
  - ...we say that advisor_id is the determinant in that functional dependency.

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- and, when you use the notation
  - A -> B
  - ...we say that A is the DETERMINANT in that functional dependency
- Note: you can have sets of attributes on either side or both sides of a functional dependency!

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- and, when you use the notation
  - A -> B
  - ...we say that A is the DETERMINANT in that functional dependency
- Note: you can have sets of attributes on either side or both sides of a functional dependency!
  - (stu_id, class_id) -> class_grade
  - advisor_id -> (advisor_lname, advisor_fname)
  - (stu_id, class_id) -> (class_grade, completion_date)

# Intro to the RELATIONAL model

- FUNCTIONAL DEPENDENCY
- make sure this makes sense:
  - If
  - A -> (B, C), it is also the case that A -> B and A -> C

# Question:

If (D, E) -> F, is it also the case that D -> F or E -> F?

1) True
2) False