Introduction to the Relational Model

- The relational database model was first proposed in a 1970 article by E.F. Codd, an IBM Fellow
- In this lecture we will cover the basic notions of relational databases
- In the relational model, data is stored in what Codd called “relations,” which you should remember from CS135.

Let's look at the Database Schema Document

- Each of the rectangles is called a "relation schema" or "table schema."
- The six table schemas taken together are called a "relational database schema," in this case for a database that will hold information about:
  - A companiy's EMPLOYEEs
  - The companiy's DEPARTMENTs
  - The LOCATIONs (offices or installations) of the company's departments
  - The PROJECTs being worked on by the company
  - Information about which employees of the company WORK ON which projects
  - Information about the DEPENDENTs of company employees

Note that

- Each table schema looks like a record/struct type declaration – which is essentially what it is
- The named items in each rectangle, for example, PNAME, MINIT, LNAME, etc. in EMPLOYEE, look like record/struct field declarations – which they essentially are. (In relational database terminology they're referred to as "attributes" rather than "fields.")
- The six schemas/declarations are the database designer's way of saying that: (see next 6 slides):

Each row of the EMPLOYEE table will hold

- *The* first name (FNAME) of *an* EMPLOYEE
- *The* middle initial (MINIT) of *that* EMPLOYEE
- *The* last name (LNAME) of *that* EMPLOYEE
- *The* social security number of *that* EMPLOYEE
- *The* birth date (BDATE) of *that* EMPLOYEE
- *The* address (ADDRESS) of *that* EMPLOYEE
- *The* sex (SEX) of *that* EMPLOYEE
- *The* salary (SALARY) of *that* EMPLOYEE
- *The* social security number of *the* supervisor (SUPERSSN) of *that* EMPLOYEE
- *The* number of *the* department (DNO) for which *that* employee works

NOTE:

Pay Attention to Italicized Words Like “a,” “an,” “the,” and “that” in Definitions of Attributes on the Previous Slide and on the Next Five Slides. In fact, let’s go back to the previous slide and see if you can explain why, for each occurrence of an instance of “a,” “an,” “the,” or “that,” the particular one on the four words was used – rather than any of the other three. ***

Each row of the DEPARTMENT table will hold

- *The* name (DNAME) of *a* DEPARTMENT
- *The* number (DNUMBER) of *that* DEPARTMENT
- *The* social security number of *the* manager (MGRSSN) of *that* DEPARTMENT
- *The* date (MGRSTARTDATE) on which *the* manager of *that* DEPARTMENT began to manage *that* DEPARTMENT

Each row of the DEPT_LOCATIONS table will hold

- *The* number (DNUMBER) of *a* DEPARTMENT
- *A* location (of *an* office or of *an* installation) of *that* DEPARTMENT

Each row of the PROJECT table will hold

- *The* name (PNAME) of *a* PROJECT
- *The* number (PNUMBER) of *that* PROJECT
- *The* location (PLOCATION) of *that* PROJECT
- *The* number of *the* department (DNUM) that controls *that* PROJECT

Each row of the WORKS_ON table will hold

- *The* social security number (ESSN) of *an* employee
- *The* number of *a* project (PNO) that *that* employee WORKS_ON
- *The* number of hours per week (HOURS) that *that* employee WORKS_ON *that* project

Each row of the DEPENDENT table will hold

- *The* social security number (ESSN) of *an* employee
- *The* first name (DEPENDENT_NAME) of *a* dependent (DEPENDENT) of *that* employee
- *The* sex (SEX) of *that* DEPENDENT
- *The* birthdate (BDATE) of *that* DEPENDENT
- *The* relationship (RELATIONSHIP) of *that* DEPENDENT to *that* employee

Schemas

- Recall that: A table/relation *schema* is a declaration of a table/relation/record type
- Recall that: A collection of table/relation *schemas* is called a "relational database schema."

If there are schemas/declarations, then there must be instances, so let's look at the Database Instance Document

- If you look at the Database Instance, you'll find an instance of each of the six schemas -- six tables/relations
- The collection of six table/relation instances constitutes an "instance of the relational database schema."

Note

- According to the mathematical definition of "relation," a relation is a *set* of tuples. (An n-attribute relation is a set of n-tuples)
- Sets don't have duplicate members
- We're using "table" for what Codd called a "relation," so, at least for now, we'll consider a table to be a *set* of rows
- This means that a table doesn't (can't) have any duplicate rows and that the order of rows of a table has no significance

Now

- Spend five minutes looking over the table *instances* shown in the posted Database Instance document to see if everything looks right about the contents of the rows of the tables…and let me know if you think there's anything that doesn't! ***

After you've looked at the table instances, look again at the table schemas

- Notice that in each schema one or more of the attributes is underlined
- The (singleton or larger) set of attributes underlined in a schema is called the schema's/table's *key*

So, For Example

- {SSN}is the key of EMPLOYEE. (For short we'll often simply say that "SSN is the key of EMPLOYEE;" that is, we'll drop the set brackets.)
- {DNUMBER} is the key of DEPARTMENT (For short we'll often simply say that "DNUMBER is the key of DEPARTMENT.")
- {DNUMBER, DLOCATION} is the key of DEPT_LOCATIONS (For short we'll often simply say that "DNUMBER and DLOCATION together are the key of DEPT_LOCATIONS.")

…and

- {PNUMBER} is the key of PROJECT (For short we'll often simply say that "PNUMBER is the key of PROJECT.")
- {ESSN, PNO} Is the key of WORKS_ON (For short we'll often simply say that "ESSN and PNO together are the key of WORKS_ON.")
- {ESSN, DEPENDENT_NAME} is the key of DEPENDENT (For short we'll often simply say that "ESSN and DEPENDENT_NAME together are the key of DEPENDENT.")

If you think you know what it means for a set S of attributes to be a key of a table schema T, write down a definition. *

Keys

- If you wrote down the definition of "key," it was probably something like "The attributes in S uniquely identify members of T."
- While that's sort of true, it's a bit imprecise: What does "uniquely identify" mean, exactly?
- The actual, operational, definition of "key" is: S is a key of table schema T means that an *instance* of T is a *legal* instance only if no two rows agree on the values of all the attributes in S. (Recall that a table isn't allowed to have duplicate rows.)

So

- A “key” is a constraint, on a table schema, imposed by the database designer
- Go back to the database instance and make sure that each of the six table *instances* is legal in the sense that it conforms to the key constraint for the relevant table *schema*

And consider the following question:

- Is it always/sometimes/never possible to determine what the key of a table schema is simply by looking at a table instance, that is without having seen the schema? ***

Keys Can Say More Than You Might Realize: Among Other Things, the Fact That SSN is the Key for EMPLOYEE Means That

- An EMPLOYEE *can't* work for more than one department; that is, an EMPLOYEE can't be shared by departments. (Not necessarily true for all companies, but true, according to the database designer, for the company for which this database was designed.)
- An EMPLOYEE *can't* have more than one supervisor; that is, an EMPLOYEE can't be supervised by a group/ committee of supervisors. (ditto)
- An EMPLOYEE *can't* have more than one FNAME, MINIT, LNAME, ADDRESS, SEX, etc., but these are probably no surprise ***

And the Fact That DNUMBER is the Key for DEPARTMENT Means That

- A DEPARTMENT can't have multiple names
- A DEPARTMENT can't be managed by a group of (more than one) people

And the Fact That DNUMBER and DLOCATION together are the Key for DEPT_LOCATIONS Means That

- A department *can* have multiple locations (DLOCATIONs)
- A location (city) *can* be the location (DLOCATION) of multiple departments

And the Fact That PNUMBER is the Key for PROJECT Means That

- A PROJECT *can't* have multiple names
- A PROJECT has *at most one* location (PLOCATION)
- A PROJECT is controlled by *at most one* department

And the Fact That ESSN and PNO together are the Key for WORKS_ON Means That

- An employee *can* work on multiple projects
- A PROJECT *can* be worked on by multiple employees

And the Fact That ESSN and DEPENDENT_NAME together are the Key for DEPENDENT Means That

- An employee *can* have multiple DEPENDENTs
- A person can be the DEPENDENT of *only one* EMPLOYEE; for example, if both of a child's parents are employees of the company, there's no way, in this database schema, to indicate that the child is the dependent of both

Q. Where Do Keys Come From? Why, for example, did the database designer choose keys that impose the following constraints? ***

- An EMPLOYEE *can't* work for more than one department
- A department *can* have multiple locations (DLOCATIONs)
- A PROJECT has *at most one* location (PLOCATION)
- A person can be the DEPENDENT of *more than one* EMPLOYEE

A. The database designer:

- Interviews employees and managers of the enterprise for which the database is being designed
- Decides on:
  - what data (attributes) must be stored in the database to support the necessary (software) applications
  - the organization of attributes into tables
  - and the keys for the tables
- All based on his/her understanding of the way the enterprise operates

Relations Between Tables

- One of Codd's brilliant inventions is a way to relate table schemas to one another by indicating that an attribute of one table is "the same attribute as" an attribute of another table.
- **So let's have a look at the DB Schema With FKRs document.**

The arrow from:

- ESSN of DEPENDENT to SSN of EMPLOYEE is called a "foreign key reference." It means that ESSN of DEPENDENT is "the same attribute" as SSN of EMPLOYEE; that is, ESSN is the social security number of an employee
- PNO of WORKS_ON to PNUMBER of PROJECT is a "foreign key reference." It means that PNO of WORKS_ON is "the same attribute as" PNUMBER of PROJECT; that is, PNO is the project number of a project
- Etc., etc., etc.

Foreign key references, like ordinary keys impose constraints. For example, the foreign key reference:

- from ESSN of DEPENDENT to SSN of EMPLOYEE means that an ESSN value may not occur in a row of DEPENDENT unless that same value occurs in the SSN field of some row of EMPLOYEE. That is, an instance of the database schema is legal only if every dependent in the DEPENDENT table is a dependent of an employee in the EMPLOYEE table.
- from PNO of WORKS_ON to PNUMBER of PROJECT means that a PNO value may not occur in a row of WORKS_ON unless that same value occurs in the PNUMBER field of some row of PROJECT. That is, an instance of the database schema is legal only if every project that some employee works on is a project known to the database instance in the PROJECT table.
- Etc., etc., etc.

HW1

- The Airline DB schema, for which you should have a handout, is an example of a DB schema from a very popular DBMS textbook. Document the schema the way the multi-company DB is documented in Section 1.1.1 of the DK SQL Chapter – including an instance for each table. ***
- Let's have a look at the DK SQL Chapter ***