A Database: a very large, integrated collection of data. (Core of CS) DBMS: maintain and query large datasets.

– Typically models a real-world “enterprise” (or a miniworld): Entities (basketball teams, players); Relationships (x team played y on <day>).

– How to design a database? : Disadvantages: Maintenance Costs, Sharing data, Extending program and Security Drawbacks

Conceptual design

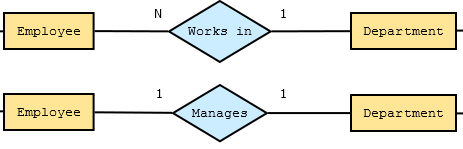
Relational model (is vast): Web (Entities: Doc; Relationships: Words, links in doc), P2P file sharing (E: filenames, hosts; R: files available)

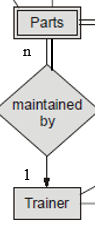
Database logical design

– How to manage and access data in DBMS? (Eg. Access control, index, transactions)

• Relational algebra

File based systems : Data stored in files, whose structure defined by programs that hold their own files (Multiple copies)

• 60’s: New proposal

– Data independent of access program

– Multiple views of data

– Data sharing

– Data monitoring, access control, etc.

• DBMS

– Allow users to create new databases and specify their schemas

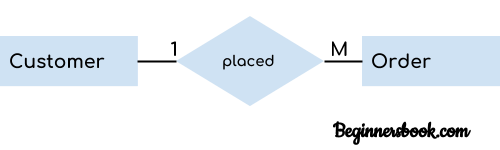
– Give users the ability to query and modify data

– Support the storage of very large amount of data

– Enable durability (data recovery is needed)

– Control access to data from many users at once

• Hierarchical model

– Each parent can have many children but each child only has

one parent

– IBM’s IMS: Apollo Project

• Network model

– Invented by Charles Bachman

BTW: he’s the first database researcher who won the ACM Turing   
– Nested-for-loop

– Problems: System complexity; Low-level DML; Difficult to maintain; No Standard; Lack of structural independence!

Awards: GE’s IDS; CODASYL (Conference on Data Systems Languages); DBTG

Codd, E. F: "A relational model of data for large shared data banks." CACM 13(6), 1970.

• Relational model

– Data independence: Separate physical implementation from logical, Data became independent from application (how it will be used)

– Data -> relations and Operations -> relational algebra or relational calculus, Set-at-a-time vs. record-at-a-time:

– Declarative language.

>CODASYL (Too complicate, Can't change set-oriented queries, No formal foundation) vs

>Relational camp (Performance, Too mathematical, Applications want record level operations)

RDBMS prototypes

– System R by IBM: RDS (relational data system) + RSS (... storage ...) OR DB2, Oracle

– key concepts: SQL (24-page paper to a 1600-page ISO standard...), query processing, views, B-tree indexing, transaction management, access control

SQL : Query Language (SEQUEL, 1974). Renamed to SQL. Ingres implements Quel, System R implements Structured English; Oracle V2: first commercially available SQL

Database- large, integrated collection of data | Data- Group of facts; can be recorded; have an implicit meaning | DBMS: Create new databases and specify their schemas

• Conceptual design: Database Management System (DBMS) is collection of programs; enable users create and maintain databases.

– “Sketch the design”

– Very subjective

– Some design principles

• Schema design:

– Rules to follow

– Normalization

ER Model: most widely used; describe a physical world; Intuitive and convenient; High level model; entity-relationship diagrams; sketch database schema

– Entity sets - Entity: A specific object or “thing” in the mini-world. degree of a relationship type = number of participating entity sets

– Relationships - relates two or more distinct entities with a specific meaning ; same type are grouped into a relationship type; can have attributes attached

• ER Diagram: Sketching the key components is an efficient way

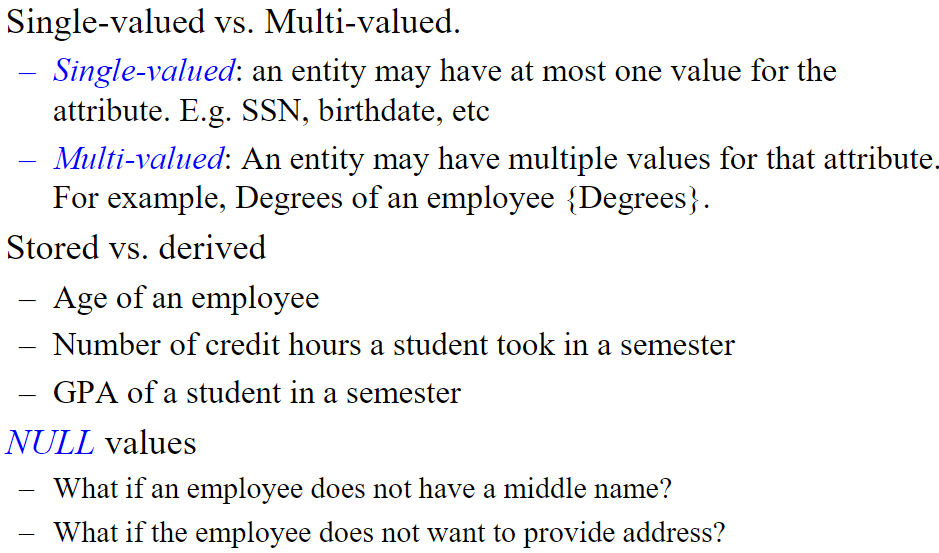
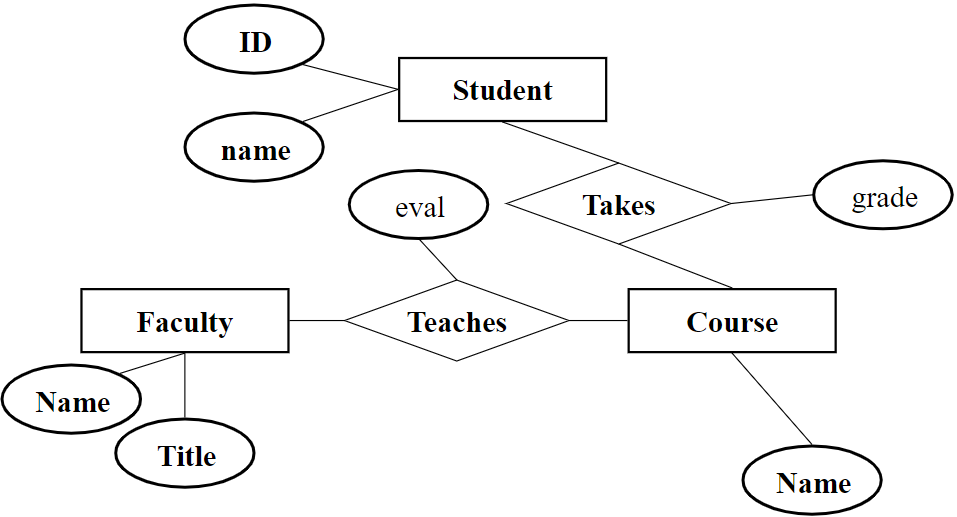
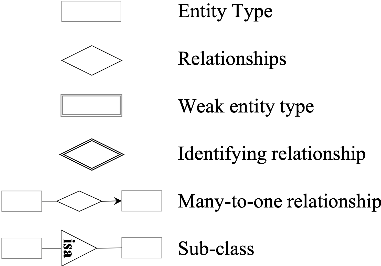
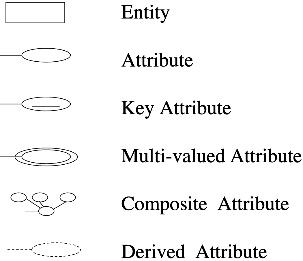
– Entity sets - collection of similar entities

– Attributes - property of/used to describe (the entities of) an entity set; are simple values, e.g. integers or character

• Keys: each entity must have unique value is called key attribute. key attribute may be composite. entity type may have 1+ key.

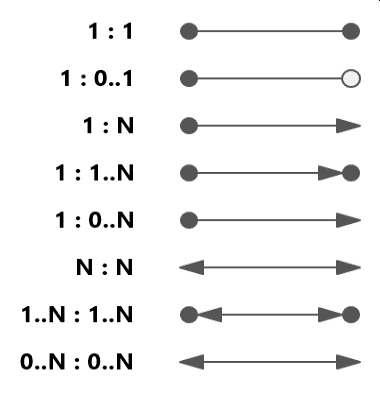
– Sub-classes (Simple: Each entity has a single atomic value, Composite: Attribute composed of several components) = special case = fewer entities = more properties

– Weak entity sets



Maximum Cardinality : Wherever the foreign key is, that's the many sign. Invoice (1) -> (M) Products [Set in one relation used to “refer” to tuple in another relation.]

– Many-to-one (N:1): Every set 1 entity connected (at most) one entity of set 2. But entity of set 2 can be to 0, 1, or ">" of first set.

– One-to-many (1:N) or – One-to-one (1:1)

– Many-to-many :

• In a many-many relationship: entity of either set connected to many entities of other set.

Design Principles: Keep simple! Avoid redundancy.

• Redundancy: wastes space/encourages inconsistency; Limit the use of weak entity sets.

Entity set may participate more in >1 relationship type. Eg: Roommate, Lovers, Married.

• Use Weak entity sets only when:

• There is no way to assign unique keys, e.g. there is no global authority capable of creating unique ID’s.

– Don’t use an entity set when an attribute will do.

• To be “qualified” as an entity set: E={e | e is an entity}

• Has at least one non-key attribute.

• “Many” in a many-one or many-many relationship.

Relational Model

> Simple but powerful mode; V. efficient implementations; RDBMS dominates industry; Handle very large-scale data

> Values are considered atomic; NULL value is used to represent unknown or inapplicable to tuples.

• What is a Data Model? Relational database: A set of relations

• Structure of Data: Mathematical representation of data; semi-structured model = trees/graphs.

Operations on data; Constraints; Examples: relational model = tables.

**Informal Terms : Formal Terms**

Table : Relation (tuples or records) | Column : Attribute (cols of relation are named by attributes)

Row (rows = cardinality) : Tuple (Excluding the header row, are not ordered) | Possible values in a column (fields = degree / arity) : Domain

Table Definition : Schema of a Relation (specifies name (relation), name & type (each column)) | Populated Table : Instances (a table, with rows and columns)

Constraints are conditions that must hold on all valid relation instances. There are three main types of constraints:

1.Domain constraints (set of allowed values is domain, value of an attribute must come from its domain)

2.Key constraints (set of rules for primary key, every relation must have a primary key, The primary key attributes can't take the NULL value)

3.Referential integrity constraints (must always exist a valid relationship between two relational database tables)

Logical Design

• An entity set translates directly to a relation

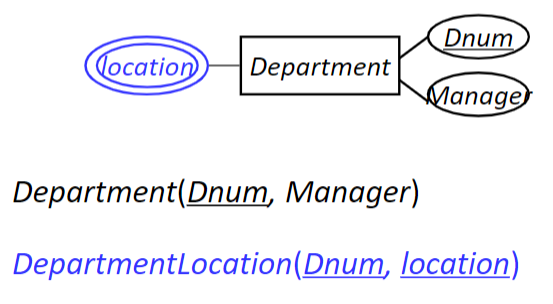
– Attributes → columns

– Key attributes → Primary key

– Composite attributes → flattened to single attributes – Multi-valued attributes → relation

– Weak entity sets: derive a key from its parent

Relationships Multivalued attributes:

Logical Design

– 1:1 relationships

• Merge into a single table: best for total participation from both sides

• Foreign Key: which direction (total participation)

• Map to a relationship relation

– 1:N relationship

• Relationship Relation

• Foreign Key: which direction? (in the bottom)

– M:N relationship :

• Why not Foreign Key? Only room for one value per cell in a relation

• Relationship Relation

– N-ary relationship

• Relationship Relation

• Normalization

– A systematic approach to: Eliminate redundancy; Avoid inconsistency

• Things to know

– Redundancy

– Functional dependency : Functional dependency (FD) has the form X →Y, when two tuples agree on attr of X, must also agree on attr of Y.

– Decomposition :

– First normal form and second normal form

1NF: Column must be single value (atomic); Remove repeating groups; All values must be same type in 1 col; Col name should be unique; No two rows can be identical

2NF: 1NF + Non-key attributes are dependent on primary key (remove all partial dependencies); tables with 1 col keys are automatically 2NF.

– Boyce-Codd normal form (BCNF), decomposition to BCNF

• Decomposition to BCNF

• Find a BCNF violation: That is, a non-trivial FD X → Y in R where X is not a super key of R

• Grow Y

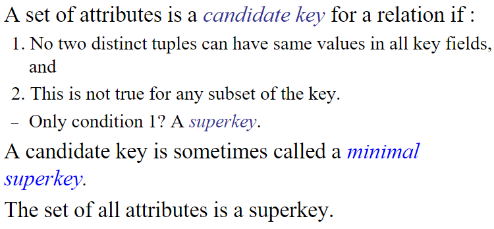
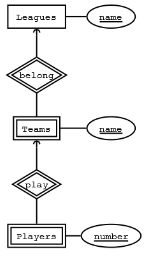
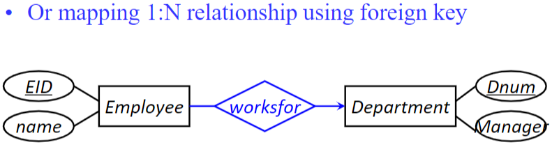
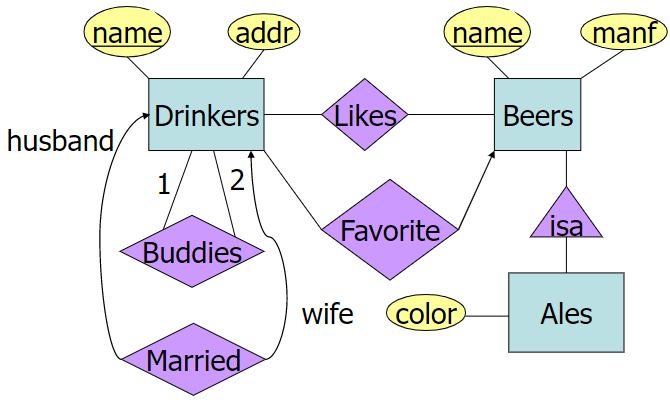
• Decompose R into R1 and R2, where

– R1 has attributes X  Y

– R2 has attributes X  Z, where Z contains all attributes of R that are in neither X nor Y (i.e. Z = attr(R) – X – Y)

• Repeat until all relations are in BCNF

EXAM:  
2.1) When a relation is known to be in first normal form, it is definitely in second normal form as well: False. When a relation in 2NF, its definitely 1NF, not vice versa.   
2.2) In a relational DBMS, data is stored in trees: False. In a relational DBMS, data is stored in two dimensional tables (relations).  
For each relational schemas and FD’s, identify ALL candidate keys, indicate FDs that violate BCNF, and decompose the relations into BCNF.   
R(A, B, C, D, E) with FDs: B→DE; B→A; A→CD; CE→B   
With B→A and A→CD, we have B→ACD   
We also know that B→DE, and B→B, hence, B→ABCDE. B is a key.   
Since CE→B→ABCDE, CE is also a key.   
A→CD is a violation. We can decompose R into: R1(A, C, D), and R2(A, B, E).

AB → C, C → D, hence: AB → A, B, C, D. Therefore: AB is a key. (We can also discover that CB and DB are keys).   
C→D and D→A violates BCNF since neither C or D is a key.   
Start with C→D (and D→A), we decompose R into R1(C, D, A) and R2(B, C).   
D→A violates BCNF since D is not a key. We further decompose R1 into R3(C, D) and R4(D, A)   
Final result: R2(B, C), R3(C, D), R4(D, A)   
3.3.1 b) B→C, D; hence AB → A, B, C, D. AB is the key.   
B→C and B→D violates BCNF. Decompose into R1(B, C, D) and R2(A, B)   
3.3.1 c) AB→C, so that AB→BC, hence AB→BC→D. Therefore, AB is a key. In this same way, we can discover that AB, BC, CD, AD are all keys.   
No BCNF violation since all FD are in the form of (key)→(other attribute)   
3.3.1 d) Each of A, B, C, and D is a key. No BCNF violation.   
3.3.1 e) R(A, B, C, D, E) with AB→C, DE→C, B→D With AB→C and B→D, we have AB→C, D; and then ABE → A, B, C, D, E. Therefore, ABE is the key.   
AB→C, DE→C, and B→D are all BCNF violations.   
If we start from AB→C, we will get: R1(A, B, C) and R2(A, B, D, E).   
B→D is still a BCNF violation in R2. We further decompose it into: R3(A, B, E) and R4(B, D).   
Final result: R 1(A, B, C), R3(A, B, E) and R4(B, D).

