Data Modeling

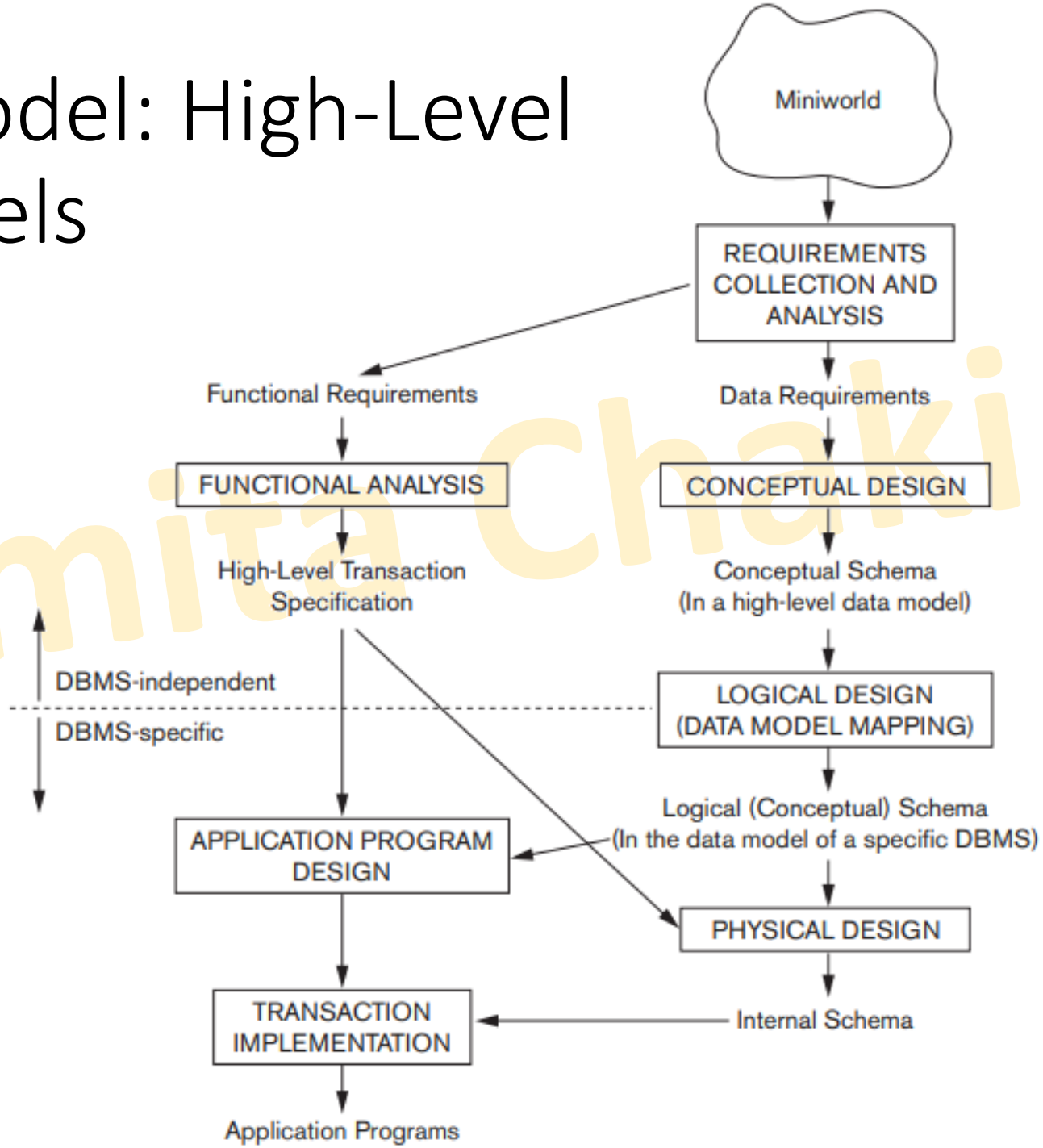
Dr. Jyotismita Chaki

Entity Relationship Model

- **Entity–relationship (ER) model** is a popular high-level conceptual data model.
  - Conceptual modeling is a very important phase in designing a successful database application.
  - Generally, the term database application refers to a particular database and the associated programs that implement the database queries and updates.
- ER model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts.
- The diagrammatic notation associated with the ER model, known as ER diagrams

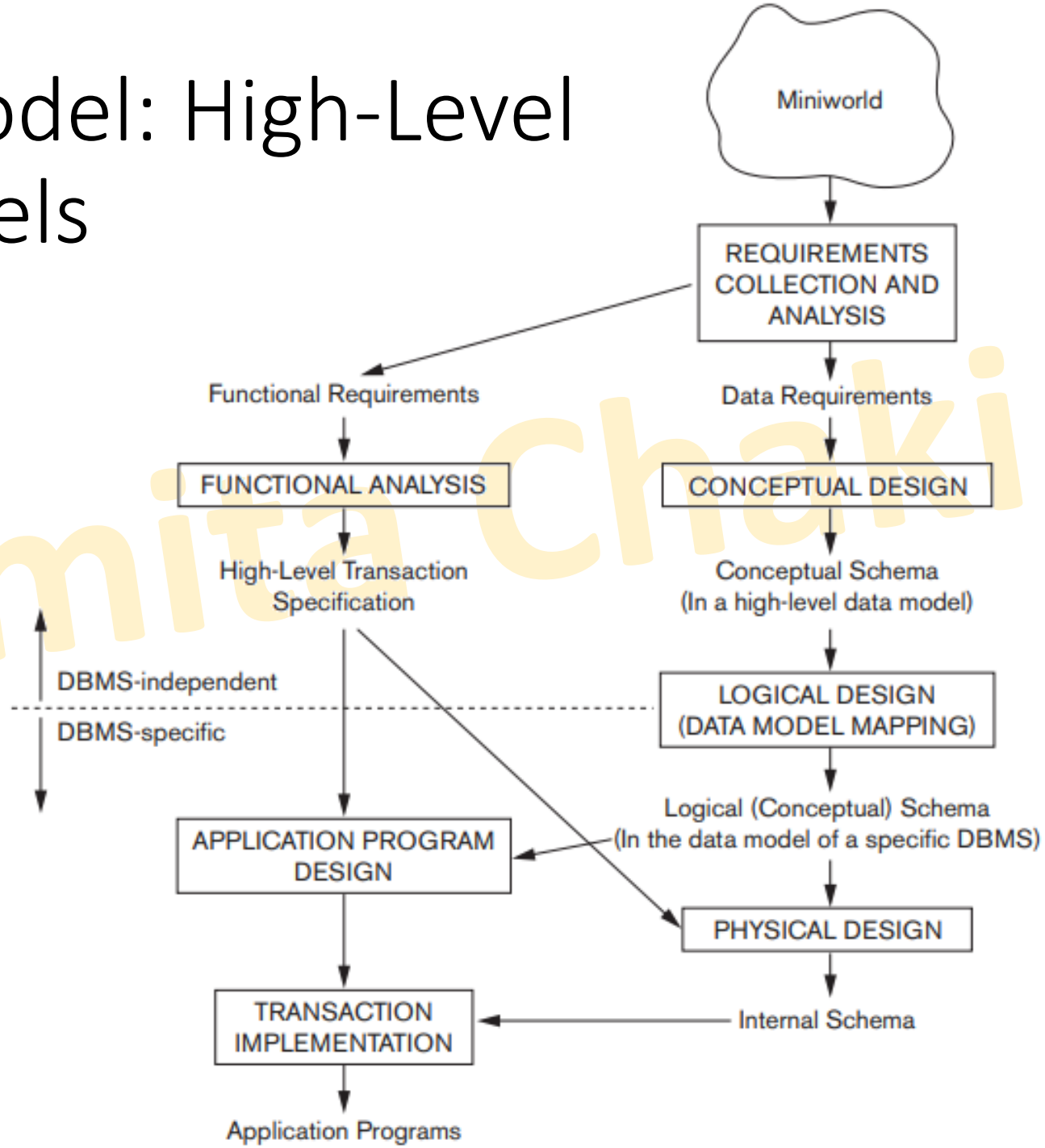
Entity Relationship Model: High-Level Conceptual Data Models

- Requirements collection and analysis: the database designers interview prospective database users to understand and document their **data requirements**
- **Functional requirements**: user defined operations (or transactions) that will be applied to the database
- Conceptual design: create a conceptual schema is a concise description of the data requirements of the users



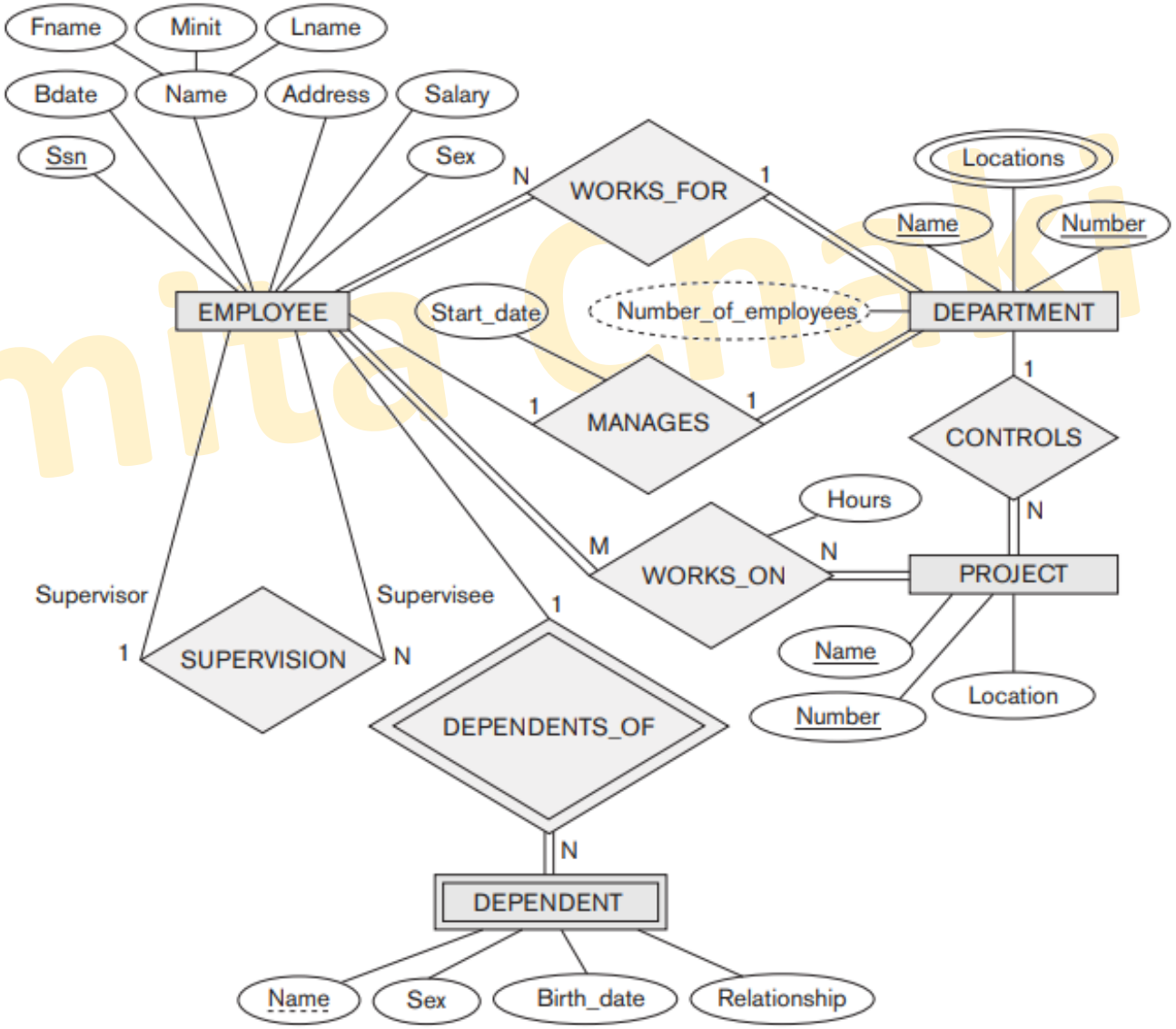
Entity Relationship Model: High-Level Conceptual Data Models

- **Logical design or data model mapping**: the conceptual schema is transformed from the high-level data model into the implementation data model.
- **Physical design**: the internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified.



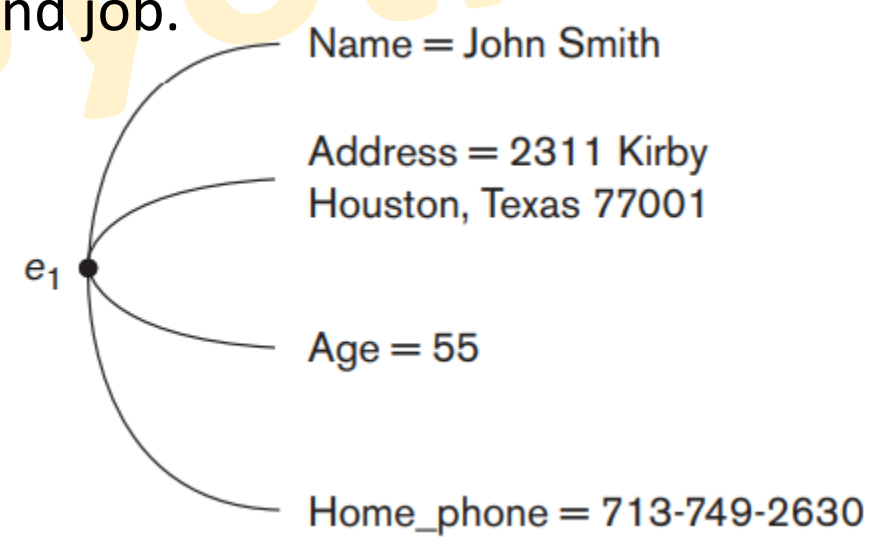
Entity Relationship Model: Example

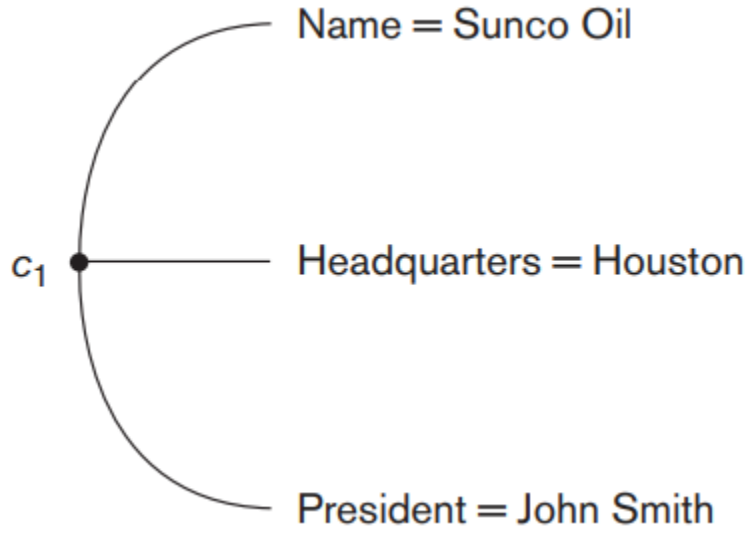
- COMPANY database keeps track of a company's employees, departments, and projects.
- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number,2 address, salary, sex (gender), and birth date.
- An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department.



Entity Relationship Model: Entities and attributes

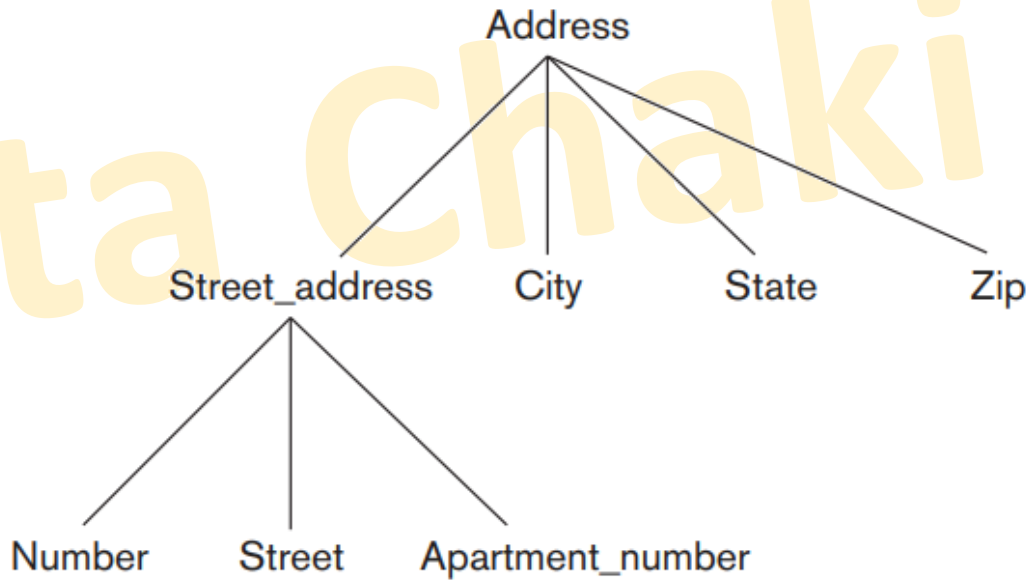
- The ER model describes data as
  - Entities: a thing or object in the real world with an independent existence. For example, a particular person, car, house, or employee.
  - Relationships, and
  - Attributes: the particular properties that describe it. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.





Entity Relationship Model: Types of attributes: Simple versus Composite

- **Composite attributes** can be divided into smaller subparts, which represent more basic attributes with independent meanings.
- Attributes that are not divisible are called **simple or atomic attributes**.
- If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes.
  - If there is no need to refer to the individual components of an address (Zip Code, street, and so on), then the whole address can be designated as a simple attribute.



Entity Relationship Model: Types of attributes: Single valued versus Multivalued

- Most attributes have a single value for a particular entity; such attributes are called **single-valued**.
  - Age is a single-valued attribute of a person.
- When there are possibilities of multiple values for a particular entity; such attributes are called **multi-valued.**
  - A Colors attribute for a car, or A College_degrees attribute for a person.
- A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.
  - The Colors attribute of a car may be restricted to have between one and two values

Entity Relationship Model: Types of attributes: Stored versus Derived

- In some cases, two (or more) attribute values are related—for example, the Age and Birth_date attributes of a person.
- The Age attribute is called a **derived attribute** and is said to be derivable from the Birth_date attribute, which is called a **stored attribute**.

Entity Relationship Model: Types of attributes: Complex

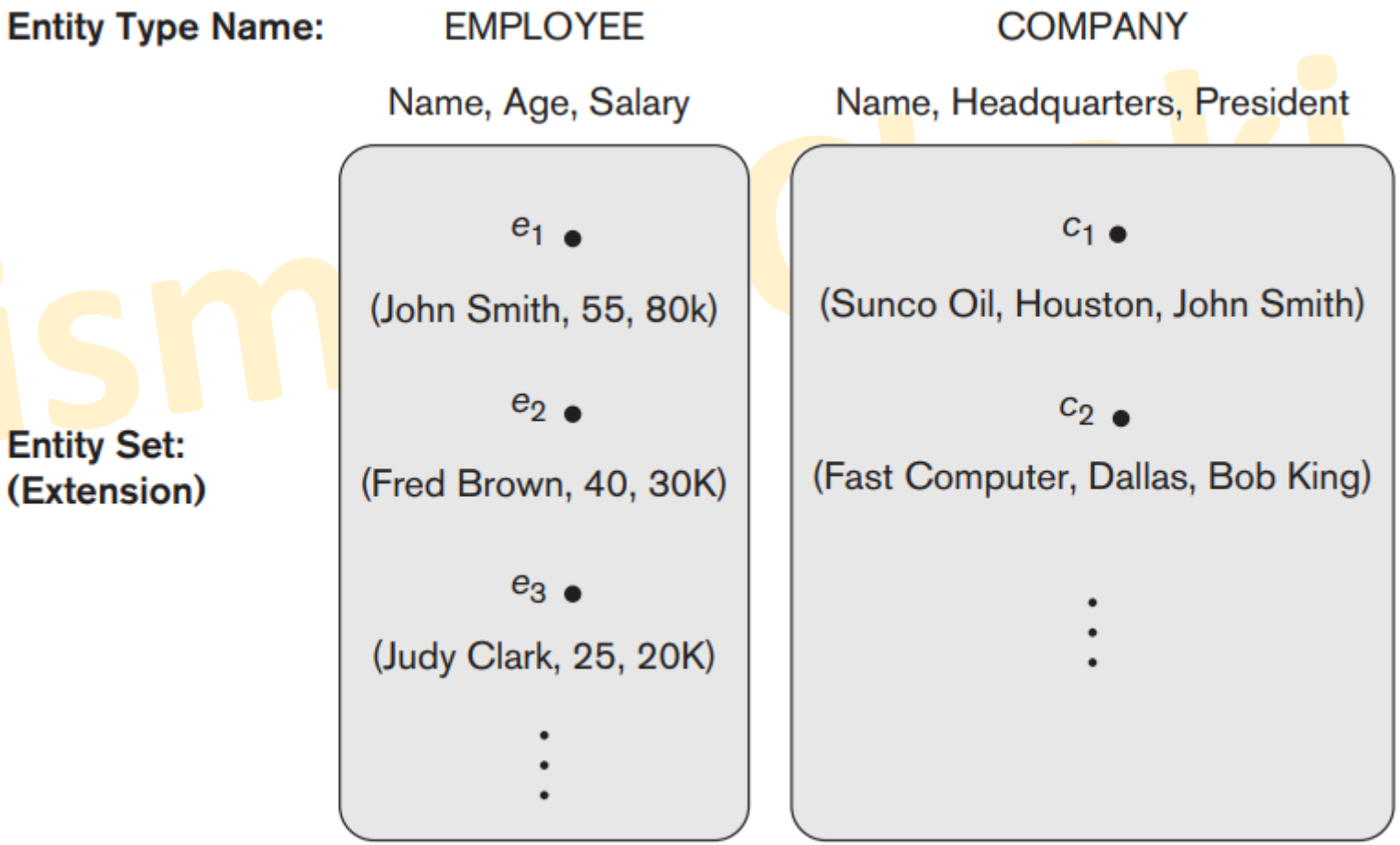
- Group of components of a composite attribute between parentheses ( ) and separating the components with commas, and by displaying multivalued attributes between braces { }.
  - {Address_phone({Phone(Area_code,Phone_number)},Address(Street_address (Number,Street,Apartment_number),City,State,Zip))}

Entity Relationship Model: Types of attributes: NULL

- In some cases, a particular entity may not have an applicable value for an attribute.
- For such situations, a special value called NULL is created.
  - A College_degrees attribute applies only to people with college degrees. A person with no college degree would have NULL for College_degrees
- NULL can also be used if we do not know the value of an attribute for a particular entity: Unknown
  - When it is known that the attribute value exists but is **missing**
  - When it is **not known** whether the attribute value exists

Entity Relationship Model: Entity Types, Entity Sets, Keys, and Value Sets

- An **entity type** defines a collection (or set) of entities that have the same attributes.
- Each entity type in the database is described by its name and attributes.
- The collection of all entities of a particular entity type in the database at any point in time is called an **entity set** or **entity collection**

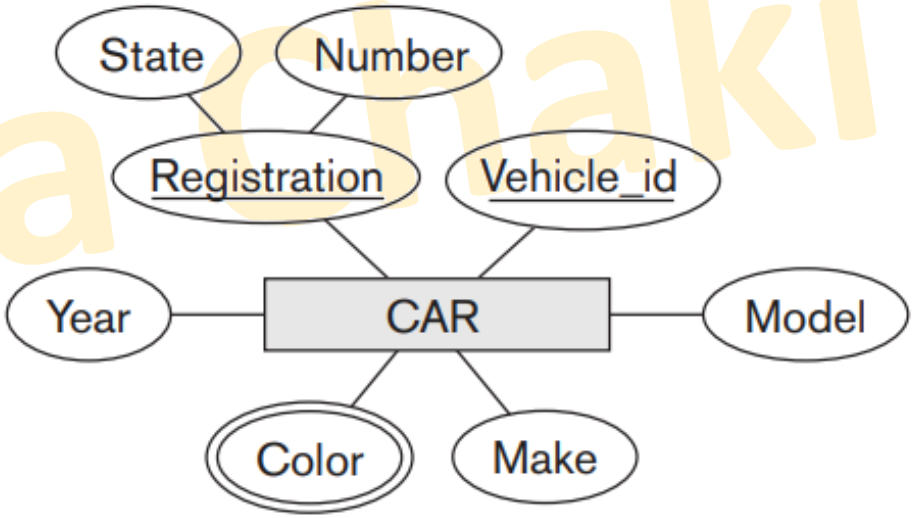


Entity Relationship Model: Entity Types, Entity Sets, Keys, and Value Sets

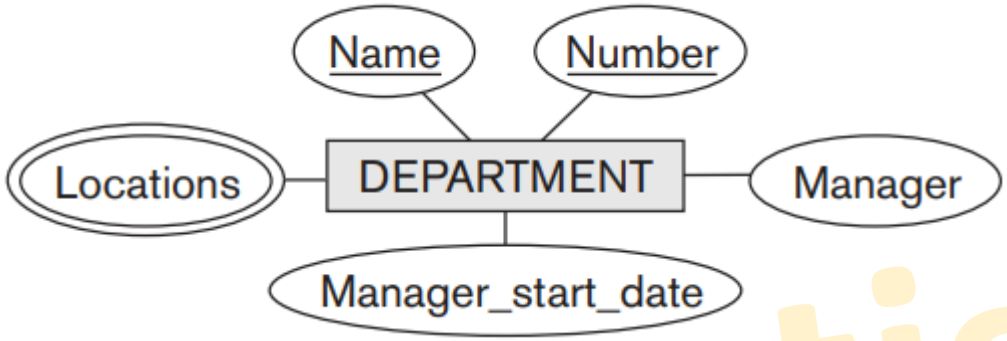
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set: **key attribute**
- Specifying that an attribute is a key of an entity type means that the preceding uniqueness property must hold for every entity set of the entity type.
- It is not the property of a particular entity set; rather, it is a constraint on any entity set of the entity type at any point in time.
- An entity type may also have no key, in which case it is called a weak entity type.
- Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity.

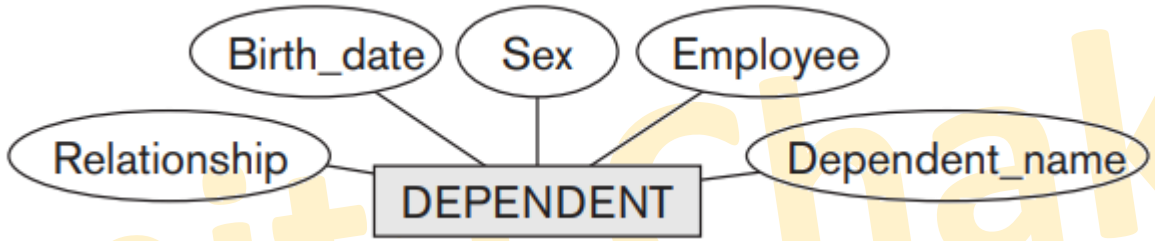
Entity Relationship Model: Notations

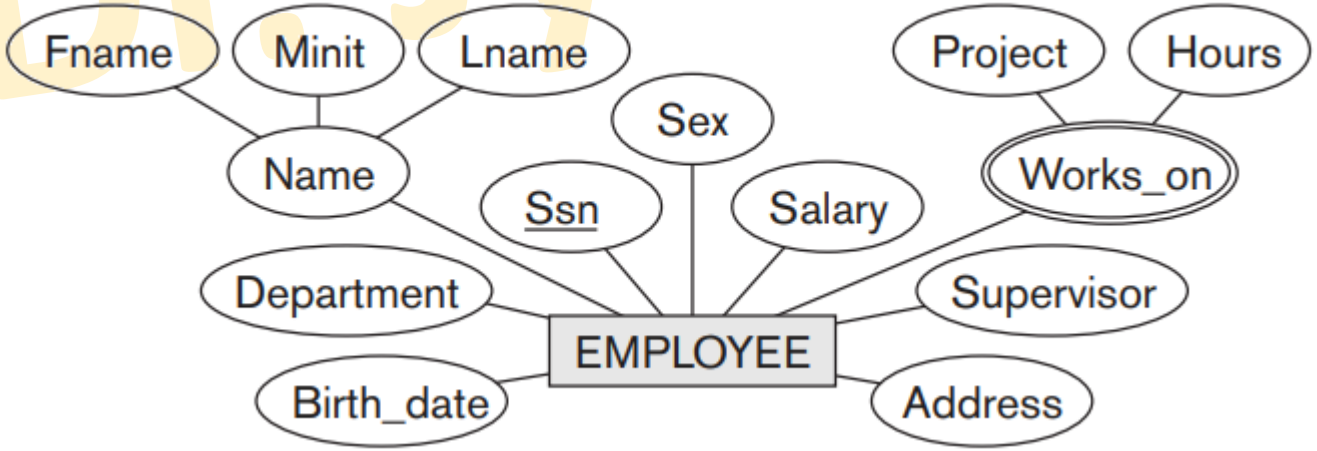
- An **entity type** is represented in ER diagrams as a rectangular box enclosing the entity type name.
- **Attribute names** are enclosed in ovals and are attached to their entity type by straight lines.
- **Composite attributes** are attached to their component attributes by straight lines.
- **Multivalued attributes** are displayed in double ovals.
- **Key attribute** has its name underlined inside the oval
- The Registration attribute is an example of a composite key formed from two simple component attributes, State and Number, neither of which is a key on its own.

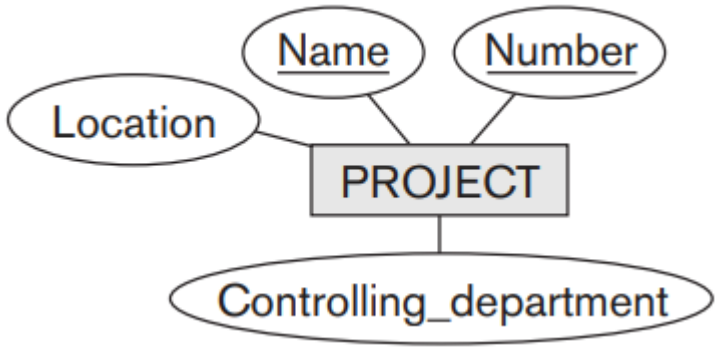


Entity Relationship Model: Example





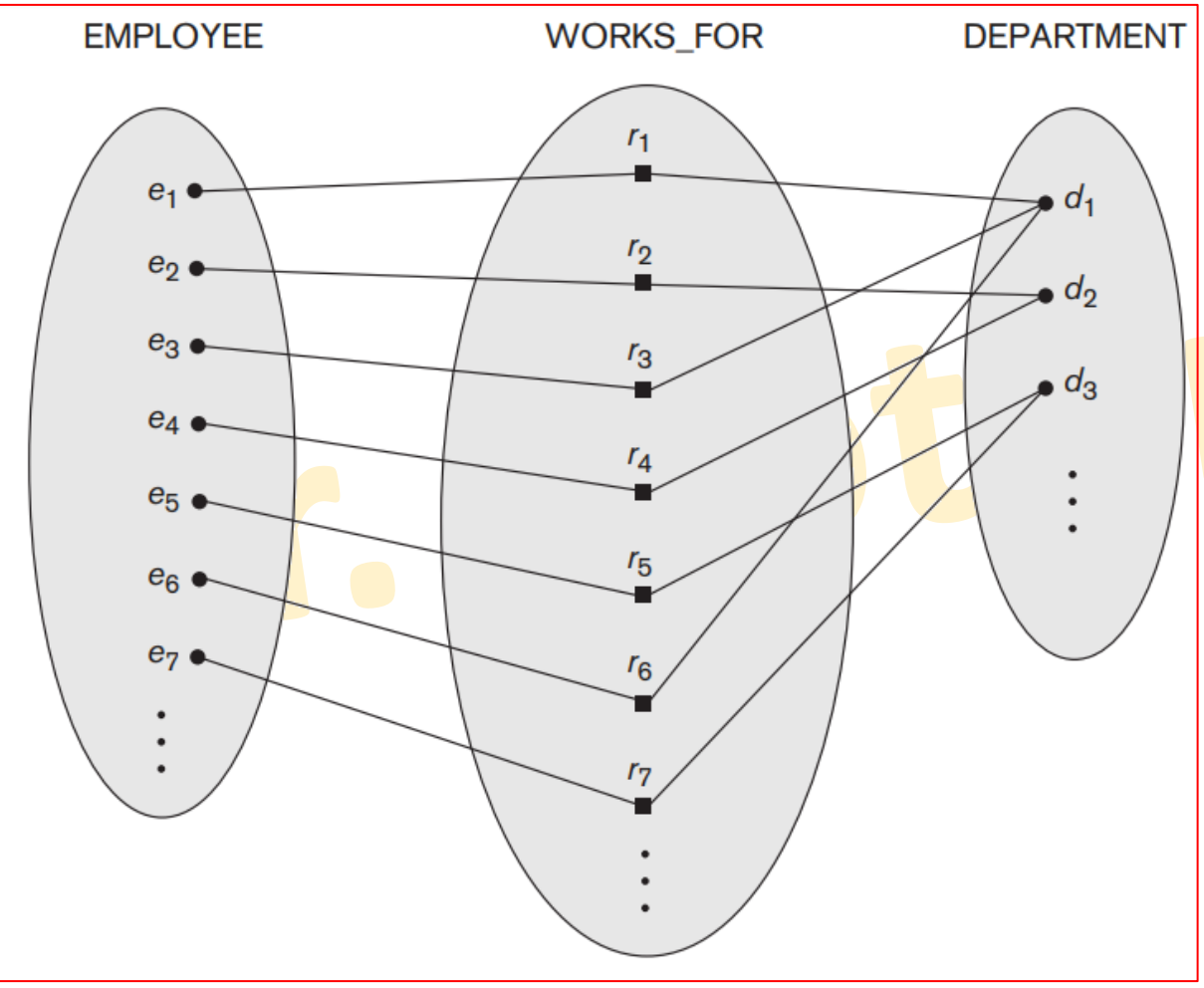


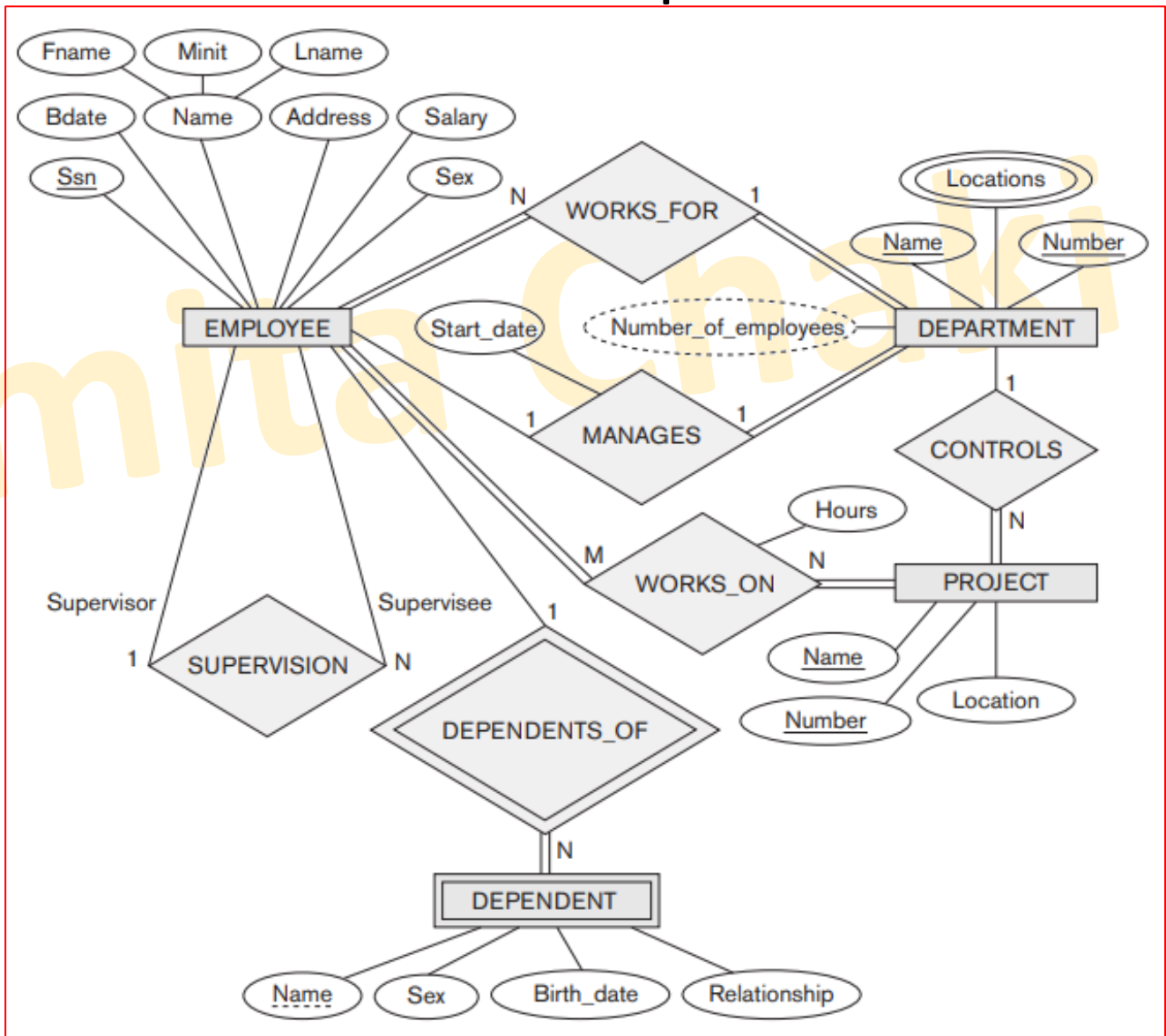


Entity Relationship Model: Relationship

- A **relationship type** R among n entity types E1, E2, . . . , En defines a set of associations—or a **relationship set**—among entities from these entity types.
- The relationship set R is a set of relationship instances ri, where each ri associates n individual entities (e1, e2, . . . , en), and each entity ej in ri is a member of entity set Ej , $1 \leq j \leq n$.
- Each of the entity types E1, E2, . . . , En is said to participate in the relationship type R; similarly, each of the individual entities e1, e2, . . . , en is said to participate in the relationship instance ri = (e1, e2, . . . , en).

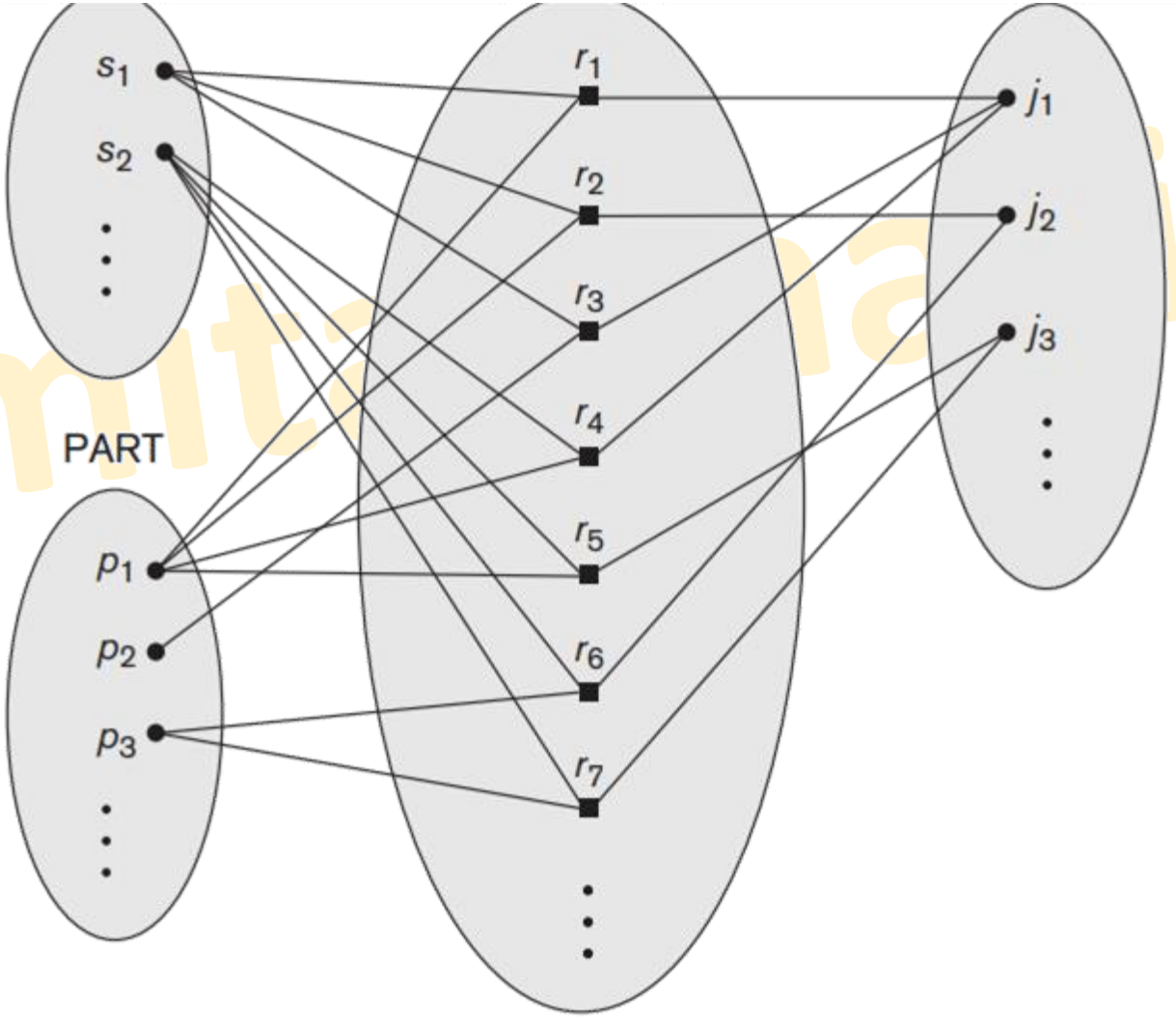
Entity Relationship Model: Relationship





Entity Relationship Model: Relationship

- The degree of a relationship type is the number of participating entity types.
- A relationship type of degree two is called binary, and one of degree three is called ternary.

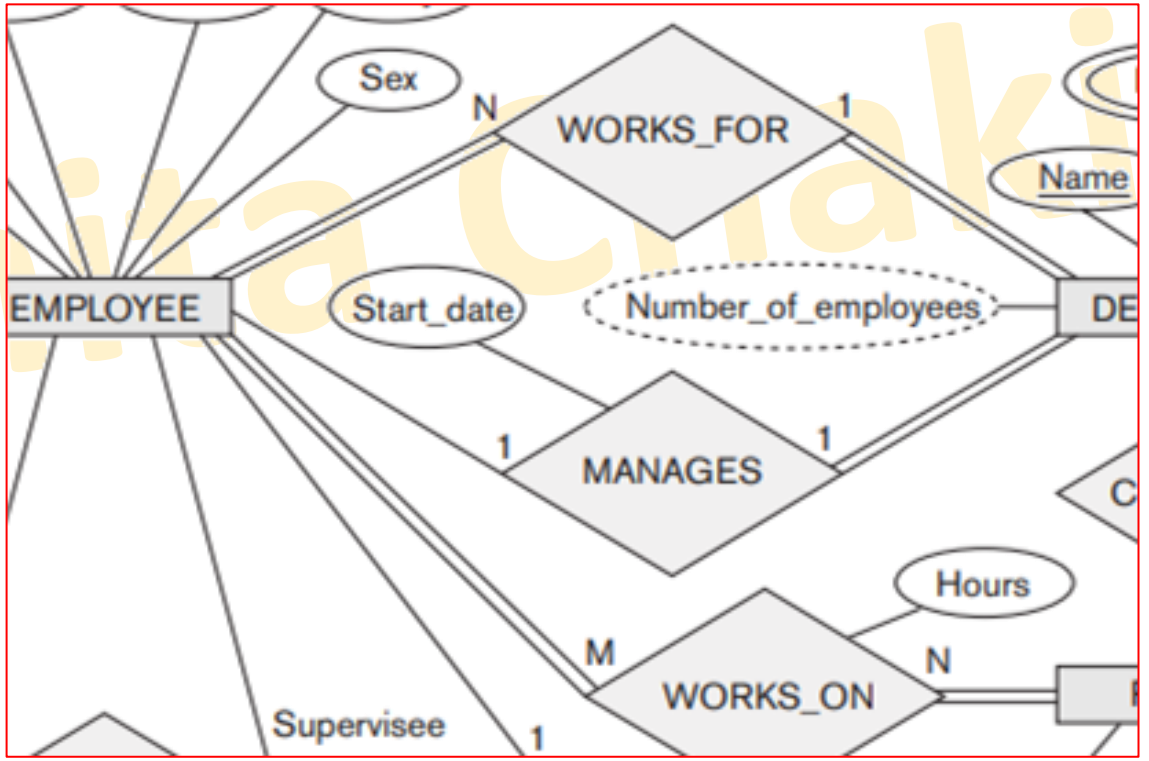


Entity Relationship Model: Binary Relationship Constraints

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.
- Two main types of binary relationship constraints:
  - **Cardinality ratio:** maximum number of relationship instances that an entity can participate in. The possible cardinality ratios for binary relationship types are 1:1 (Employee [Manages] Department), 1:N (Department:Employee), N:1 (Student:Project), and M:N (Employee [Works on] Project). Cardinality ratios for binary relationships are represented on ER diagrams by displaying 1, M, and N on the diamonds.
  - **Participation:** This constraint specifies the minimum number of relationship instances that each entity can participate in: **minimum cardinality constraint**.

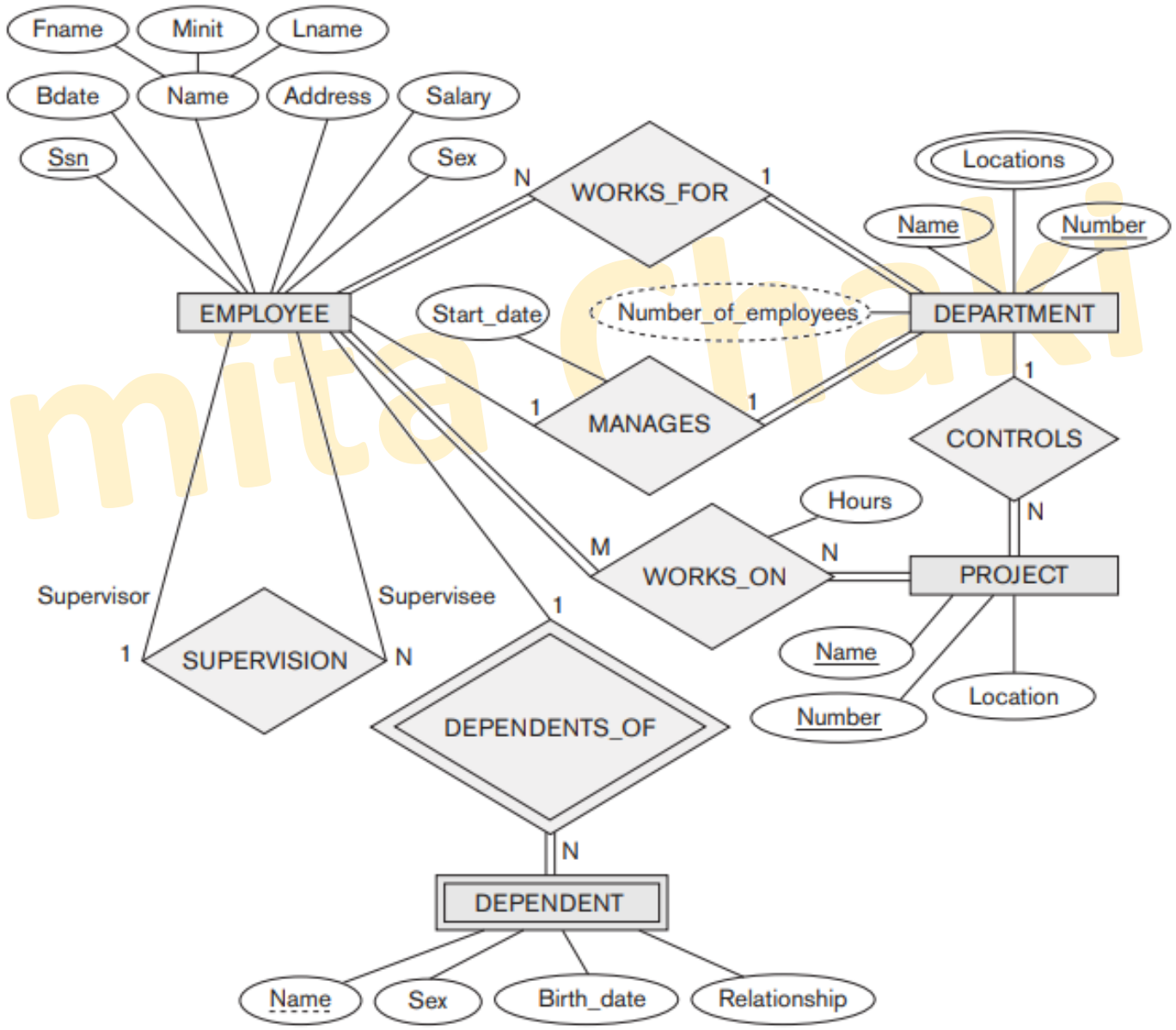
Entity Relationship Model: Binary Relationship Constraints

- Two types of participation constraints—
  - Total: If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance: **existence dependency**
  - Partial: we do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial
- In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line



Entity Relationship Model: Structural constraint

- The cardinality ratio and participation constraints, taken together, as the **structural constraints** of a relationship type

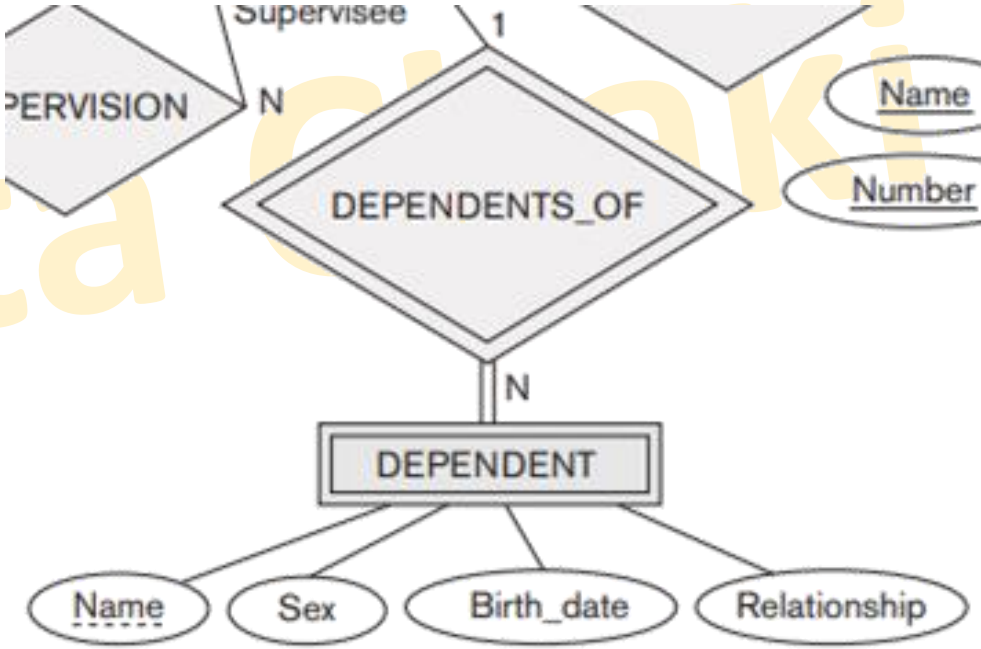


Entity Relationship Model: Weak Entity

- Entity types that do not have key attributes of their own are called weak entity types.
- Regular entity types that do have a key attribute—are called strong entity types.
- The relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type
- A weak entity type always has a total participation constraint (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.
- Not every existence dependency results in a weak entity type.
  - For example, a DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity

Entity Relationship Model: Weak Entity

- A weak entity type normally has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity.
- For example, if we assume that no two dependents of the same employee ever have the same first name, the attribute Name of DEPENDENT is the partial key.
- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line.



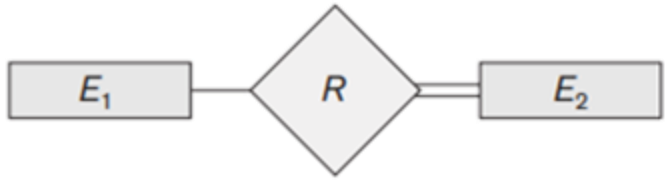
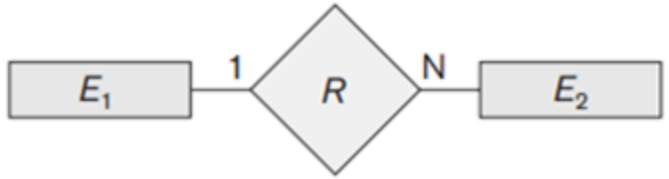
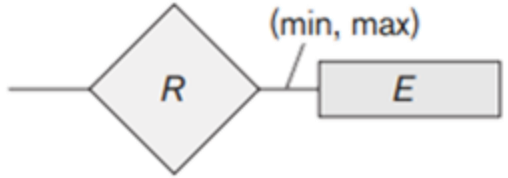
Entity Relationship Model: Summary of Notation for ER Diagrams

- Regular (strong) entity types such as EMPLOYEE, DEPARTMENT, and PROJECT are shown in rectangular boxes.
- Relationship types such as WORKS_FOR, MANAGES, CONTROLS, and WORKS_ON are shown in diamond-shaped boxes attached to the participating entity types with straight lines.
- Attributes are shown in ovals, and each attribute is attached by a straight line to its entity type or relationship type.
- Component attributes of a composite attribute are attached to the oval representing the composite attribute, as illustrated by the Name attribute of EMPLOYEE.
- Multivalued attributes are shown in double ovals, as illustrated by the Locations attribute of DEPARTMENT.
- Key attributes have their names underlined. Derived attributes are shown in dotted ovals, as illustrated by the Number_of_employees attribute of DEPARTMENT.

Entity Relationship Model: Summary of Notation for ER Diagrams

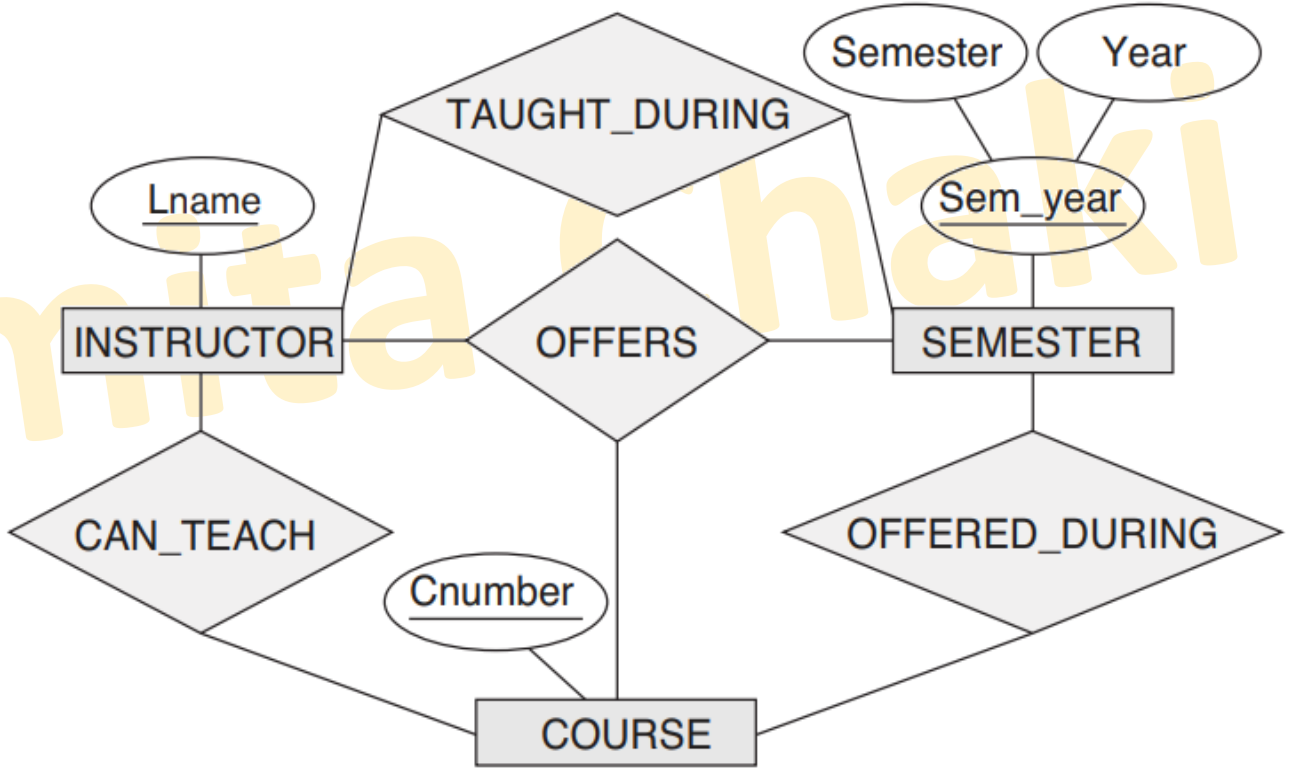
- Weak entity types are distinguished by being placed in double rectangles and by having their identifying relationship placed in double diamonds, as illustrated by the DEPENDENT entity type and the DEPENDENTS_OF identifying relationship type.
- The partial key of the weak entity type is underlined with a dotted line.
- The cardinality ratio of each binary relationship type is specified by attaching a 1, M, or N on each participating edge.
- The cardinality ratio of DEPARTMENT:EMPLOYEE in MANAGES is 1:1, whereas it is 1:N for DEPARTMENT: EMPLOYEE in WORKS_FOR, and M:N for WORKS_ON.
- The participation constraint is specified by a single line for partial participation and by double lines for total participation (existence dependency).

Entity Relationship Model: Summary of Notation for ER Diagrams

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for $E_1 : E_2$ in R
	Structural Constraint (min, max) on Participation of E in R

Entity Relationship Model: Ternary Relationship

- Relationship type of degree three
- The figure includes a relationship instance (i, s, c) whenever INSTRUCTOR i offers COURSE c during SEMESTER s.
- The three binary relationship types:
  - CAN_TEACH relates a course to the instructors who can teach that course,
  - TAUGHT_DURING relates a semester to the instructors who taught some course during that semester, and
  - OFFERED_DURING relates a semester to the courses offered during that semester by any instructor.



Relational Model

- Represents the database as a collection of relations
- A relation is thought of as a table of values
- Each row in the table represents a collection of related data values.
- The table name and column names are used to help to interpret the meaning of the values in each row
- All values in a column are of the same data type.
- A row is called a tuple, a column header is called an attribute, and the table is called a relation

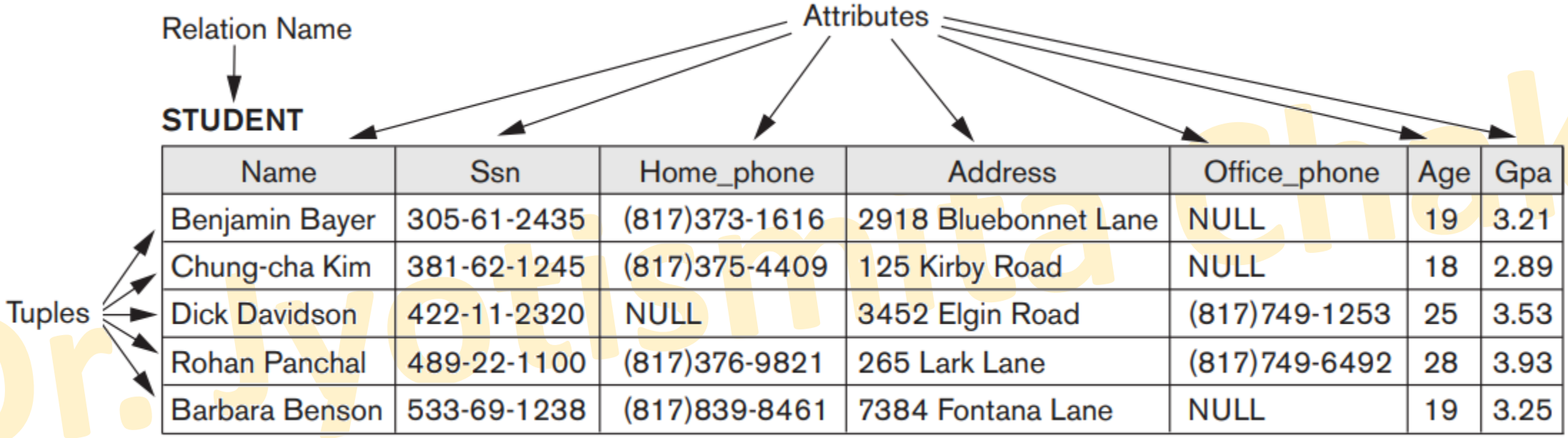
Relational Model: Domains, Attributes, Tuples, and Relations

- A **domain** D is a set of atomic values.
- By **atomic** we mean that each value in the domain is indivisible as far as the formal relational model is concerned.
- A domain is a set of acceptable values that a column is allowed to contain.
- This is based on various properties and the data type for the column.
- Ex:
  - The domain of Shift has the set of all possible days: {Mon, Tue, Wed…}.
  - The domain of Salary is the set of all floating-point numbers greater than 0 and less than 200,000.
  - The domain of First Name is the set of character strings that represents names of people.
  - Employee_ages: Possible ages of employees in a company; each must be an integer value between 15 and 80.
  - Usa_phone_numbers. The set of ten-digit phone numbers valid in the United States
- A data type or format is also specified for each domain.

Relational Model: Domains, Attributes, Tuples, and Relations

- A **relation schema** R, denoted by R(A1, A2, … , An), is made up of a relation name R and a list of attributes, A1, A2, … , An.
- D is called the domain of A_i and is denoted by **dom(A_i)**.
- A relation schema is used to *describe* a relation; R is called the name of this relation.
- The **degree (or arity)** of a relation is the number of attributes *n* of its relation schema.
- A relation of degree seven, which stores information about university students, would contain seven attributes describing each student as follows:
  - STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)
  - STUDENT(Name: string, Ssn: string, Home_phone: string, Address: string, Office_phone: string, Age: integer, Gpa: real)
  - STUDENT relation: dom(Name) = Names; dom(Ssn) = Social_security_numbers; dom(HomePhone) = USA_phone_numbers , dom(Office_phone) = USA_phone_numbers, and dom(Gpa) = Grade_point_averages.

Relational Model: Domains, Attributes, Tuples, and Relations



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

- A relation state at a given time—the current relation state—reflects only the valid tuples that represent a particular state of the real world.
- It is possible for several attributes to have the same domain.

Relational Model: Characteristics: Ordering of Tuples in a Relation

- Tuples in a relation do not have any particular order.
- Tuple ordering is not part of a relation definition because a relation attempts to represent facts at a logical or abstract level.
- Ex: (Two identical tuples when the order of attributes and values is not part of relation definition)
  - t = < (Name, Dick Davidson),(Ssn, 422-11-2320),(Home_phone, NULL),(Address, 3452 Elgin Road), (Office_phone, (817)749-1253),(Age, 25),(Gpa, 3.53)>
  - t = < (Address, 3452 Elgin Road),(Name, Dick Davidson),(Ssn, 422-11-2320),(Age, 25), (Office_phone, (817)749-1253),(Gpa, 3.53),(Home_phone, NULL)>

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relational Model: Characteristics: Values and NULLs in the Tuples

- Each value in a tuple is an **atomic** value; that is, it is not divisible into components within the framework of the basic relational model.
- Hence, composite and multivalued attributes are not allowed.
- This model is sometimes called the **flat relational model**.
- Several meanings for NULL values,
  - Value unknown,
  - Value exists but is not available,
  - Attribute does not apply to this tuple (also known as value undefined).

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relational Model: Characteristics: Interpretation (Meaning) of a Relation

- The relation schema can be interpreted as a declaration or a type of assertion.
  - For example, the schema of the STUDENT relation asserts that, in general, a student entity has a Name, Ssn, Home_phone, Address, Office_phone, Age, and Gpa.
- Each tuple in the relation can then be interpreted as a fact or a particular instance of the assertion.
  - For example, the first tuple asserts the fact that there is a STUDENT whose Name is Dick Davidson, Ssn is 422-11-2320, Age is 25, and so on.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relational Model: Characteristics: Interpretation (Meaning) of a Relation

- An alternative interpretation of a relation schema is as a predicate; in this case, the values in each tuple are interpreted as values that satisfy the predicate.
  - For example, the predicate STUDENT (Name, Ssn, …) is true for the five tuples in relation STUDENT of Figure.
- These tuples represent five different propositions or facts in the real world.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Relational Model: Notation

- A relation schema R of degree n is denoted by $R(A_1, A_2, \ldots, A_n)$.
- The uppercase letters Q, R, S denote relation names.
- The lowercase letters q, r, s denote relation states.
- The letters t, u, v denote tuples.
- An attribute A can be qualified with the relation name R to which it belongs by using the dot notation R.A—
  - For example, STUDENT.Name or STUDENT.Age.
  - Reason: The same name may be used for two attributes in different relations. However, all attribute names in a particular relation must be distinct.

Relational Model: Notation

- An n-tuple t in a relation *r(R)* is denoted by $t = <v_1, v_2, \ldots, v_n>$, where v_i is the value corresponding to attribute A_i.
- Notation for Component values of tuples:
  - Both $t[A_i]$ and $t.A_i$ (and sometimes $t[i]$) refer to the value v_i in t for attribute A_i.
  - Both $t[A_u, A_w, \ldots, A_z]$ and $t.(A_u, A_w, \ldots, A_z)$, where $A_u, A_w, \ldots, A_z$ is a list of attributes from R, refer to the subtuple of values from t corresponding to the attributes specified in the list.
    - As an example, consider the tuple t = <'Barbara Benson', '533-69-1238', '(817)839-8461', '7384 Fontana Lane', NULL, 19, 3.25> from the STUDENT relation; we have t[Name] = < 'Barbara Benson'>, and t[Ssn, Gpa, Age] = <'533-69-1238', 3.25, 19>.

Relational Model Constraints

- 3 categories:
  - **Inherent model-based constraints or implicit constraints**: Constraints that are inherent in the data model. Example: In the relational model, no two tuples in a relation can be duplicates.
  - **Schema-based constraints or explicit constraints**: Constraints that can be directly expressed in the schemas of the data model. Includes domain constraints, key constraints, constraints on NULLs, entity integrity constraints, and referential integrity constraints.
  - **Application-based or semantic constraints or business rules**: Expressed and enforced by the application programs or in some other way. Example: No employee may have a salary greater than that of her supervisor.

Relational Model Constraints: Schema-based constraints

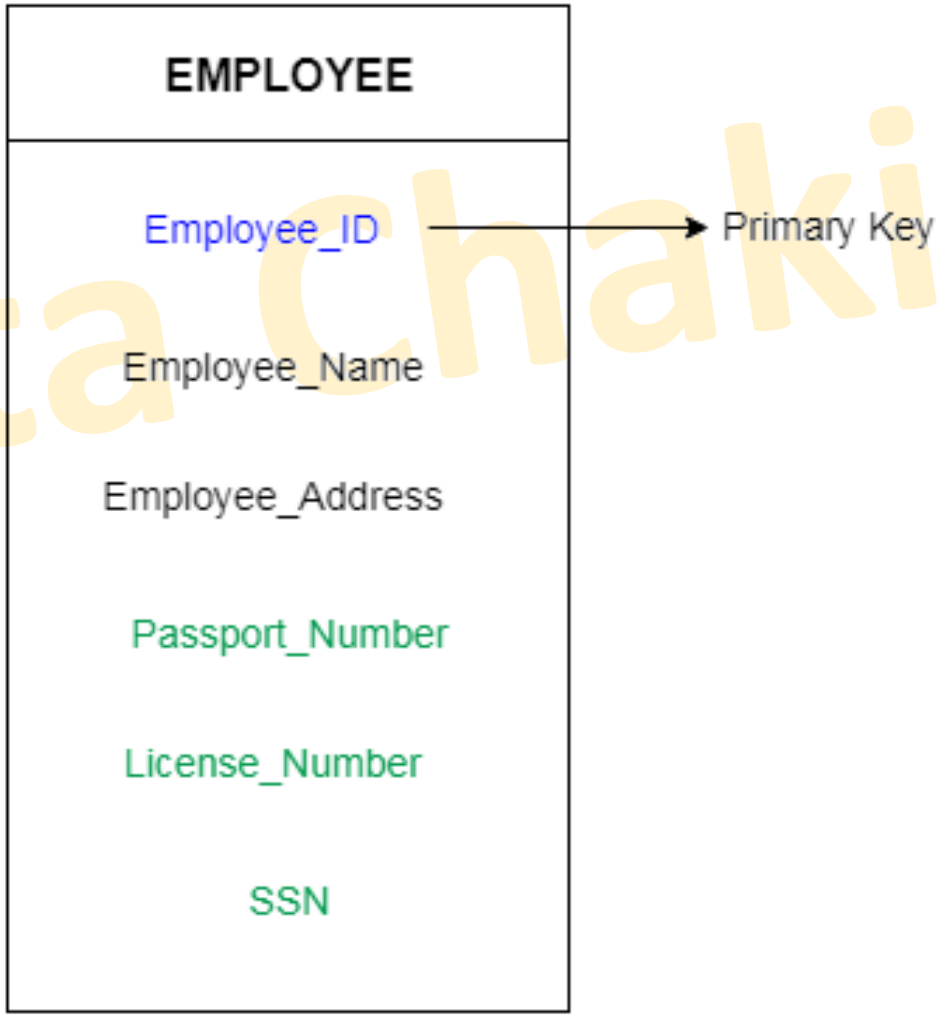
- Domain Constraints:
  - Specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).
- Key Constraints:
  - A **superkey** SK specifies a **uniqueness constraint** that no two distinct tuples in any state r of R can have the same value for SK.
  - A **key** k of a relation schema R is a superkey of R with the additional property that removing any attribute A from K leaves a set of attributes K' that is not a superkey of R any more.
  - A superkey from which we cannot remove any attributes and still have the uniqueness constraint hold - **minimal superkey.** This minimality property is required for a key but is optional for a superkey.

Relational Model Constraints: Schema-based constraints

- Key Constraints:
  - A key is a superkey but not vice versa.
  - A superkey may be a key (if it is minimal) or may not be a key (if it is not minimal).
  - Ex: The attribute set {Ssn} is a key of STUDENT because no two student tuples can have the same value for Ssn. Any set of attributes that includes Ssn—for example, {Ssn, Name, Age}—is a superkey.
  - Any superkey formed from a single attribute is also a key
  - A key with multiple attributes must require all its attributes together to have the uniqueness property.

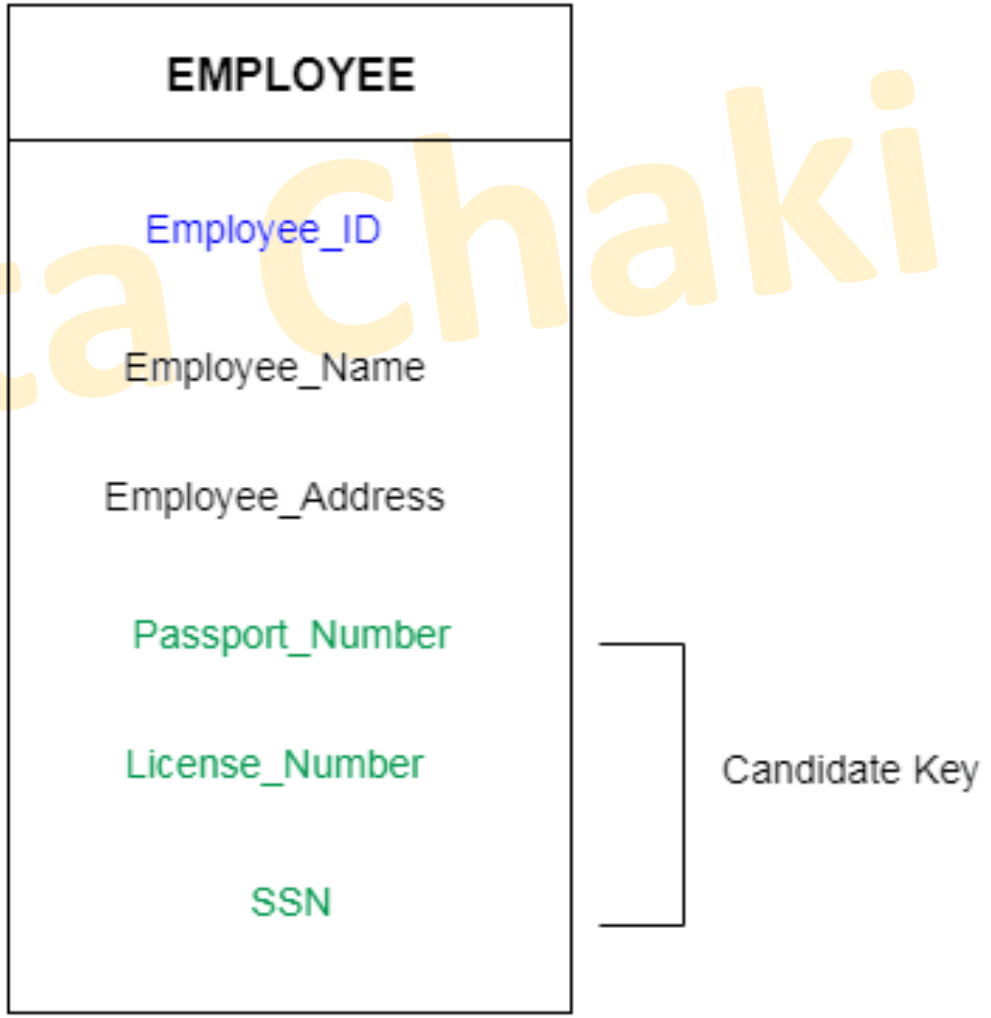
Relational Model Constraints: Schema-based constraints: Key Types

- Primary Key:
  - Used to identify one and only one instance of an entity uniquely.
  - The attributes that form the primary key of a relation schema are underlined
- Superkey
  - A set of an attribute which can uniquely identify a tuple.
  - **For example:** In the EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.



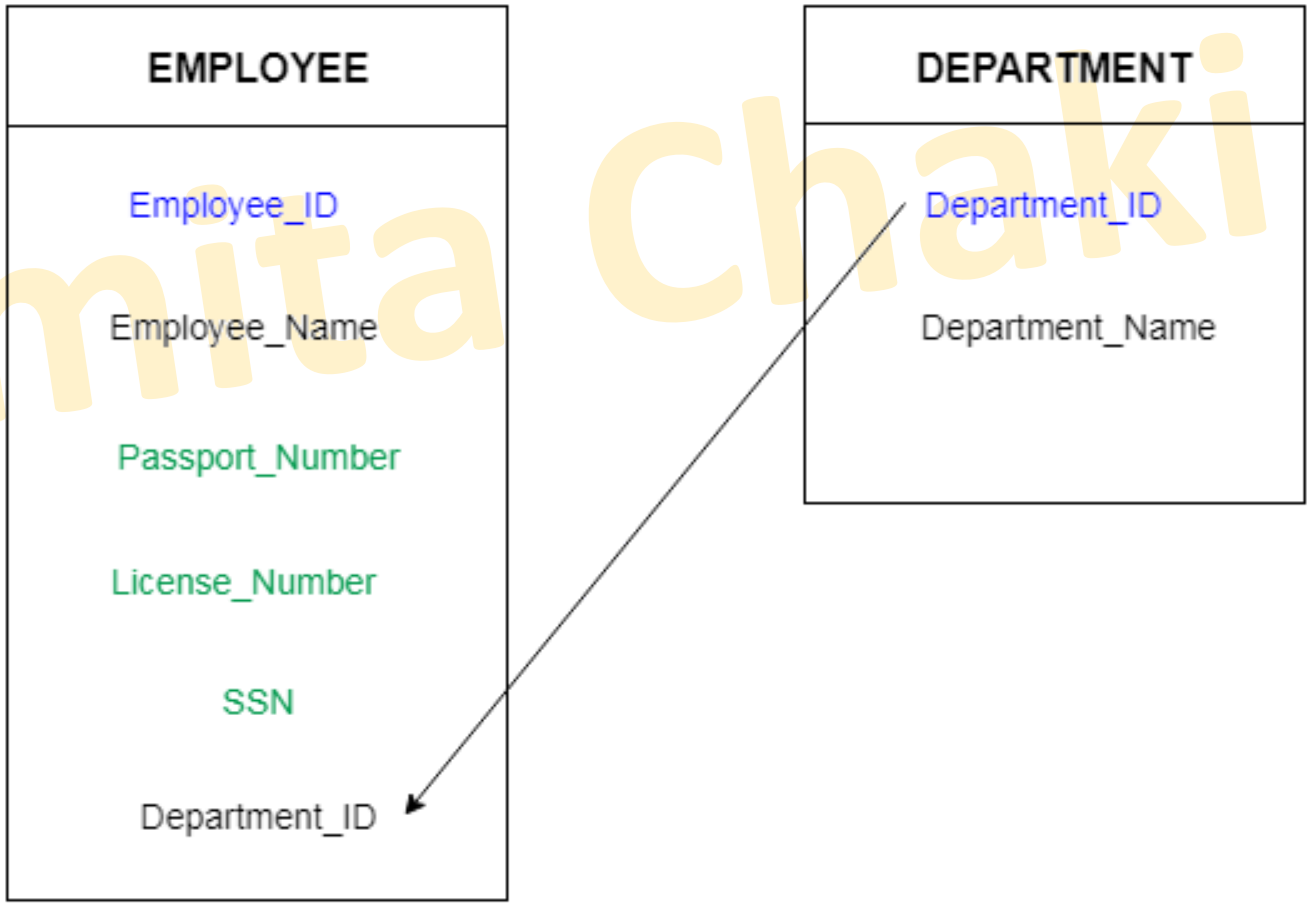
Relational Model Constraints: Schema-based constraints: Key Types

- Candidate Key
  - An attribute or set of an attribute which can uniquely identify a tuple.
  - The remaining attributes except for primary key are considered as a candidate key.
  - The candidate keys are as strong as the primary key.
  - It is usually better to choose a primary key with a single attribute or a small number of attributes. The other candidate keys are designated as unique keys and are not underlined.



Relational Model Constraints: Schema-based constraints: Key Types

- Foreign Key
  - Foreign keys are the column of the table which is used to point to the primary key of another table.
  - Eg: We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table. Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



Relational Model Constraints:

- Constraints on NULL Values:
  - Ex: If every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.
- Relational Databases and Relational Database Schemas:
  - A **relational database schema** S is a set of relation schemas $S = \{R_1, R_2, \ldots, R_m\}$ and a set of **integrity constraints** IC.
  - A **relational database state** DB of S is a set of relation states $DB = \{r_1, r_2, \ldots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.

Relational Model Constraints

- Relational database schema example:
  - COMPANY = {EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, DEPENDENT}
  - In each relation schema, the underlined attribute represents the primary key.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Relational Model Constraints

- One possible database state for the COMPANY relational database schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son

Relational Model Constraints: Integrity constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.
- Types:
  - Entity Integrity
  - Referential Integrity

Relational Model Constraints: Integrity constraints: Entity Integrity

- The entity integrity constraint states that no primary key value can be NULL.
- This is because the primary key value is used to identify individual tuples in a relation.
- Having NULL values for the primary key implies that we cannot identify some tuples.
- For example, if two or more tuples had NULL for their primary keys, we may not be able to distinguish them if we try to reference them from other relations.
- Key constraints and entity integrity constraints are specified on individual relations

Relational Model Constraints: Integrity constraints: Referential integrity

- The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.
- For example, in the following Figure, the attribute Dno of EMPLOYEE gives the department number for which each employee works; hence, its value in every EMPLOYEE tuple must match the Dnumber value of some tuple in the DEPARTMENT relation.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

DEPARTMENT

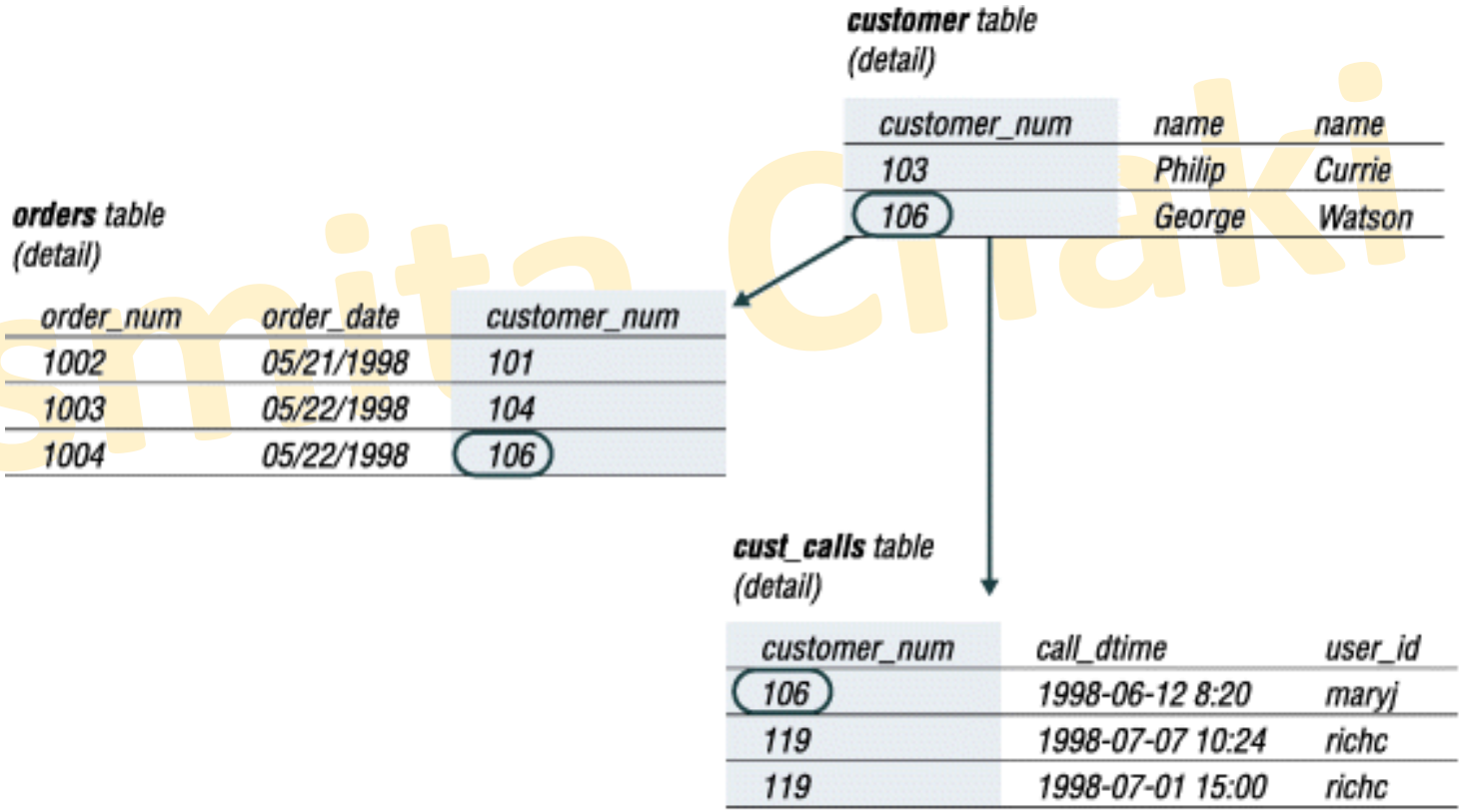
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01

Relational Model Constraints: Integrity constraints: Referential integrity

- The attributes in FK have the same domain(s) as the primary key attributes PK of R_2; the attributes FK are said to reference or refer to the relation R_2.
- $t_1[FK] = t_2[PK]$, →the tuple t_1 **references or refers** to the tuple t_2.
- In this definition, R_1 is called the **referencing relation** and R_2 is the **referenced relation**.
- If these two conditions hold, a **referential integrity constraint** from R_1 to R_2 is said to hold.

Relational Model Constraints: Integrity constraints: Referential integrity

- Foreign keys join tables and establish dependencies between tables. tables can form a hierarchy of dependencies in such a way that if you change or delete a row in one table, you destroy the meaning of rows in other tables.
- Referential integrity is the logical dependency of a foreign key on a primary key.



Relational Model Constraints: Integrity constraints: Referential integrity

EMPLOYEE

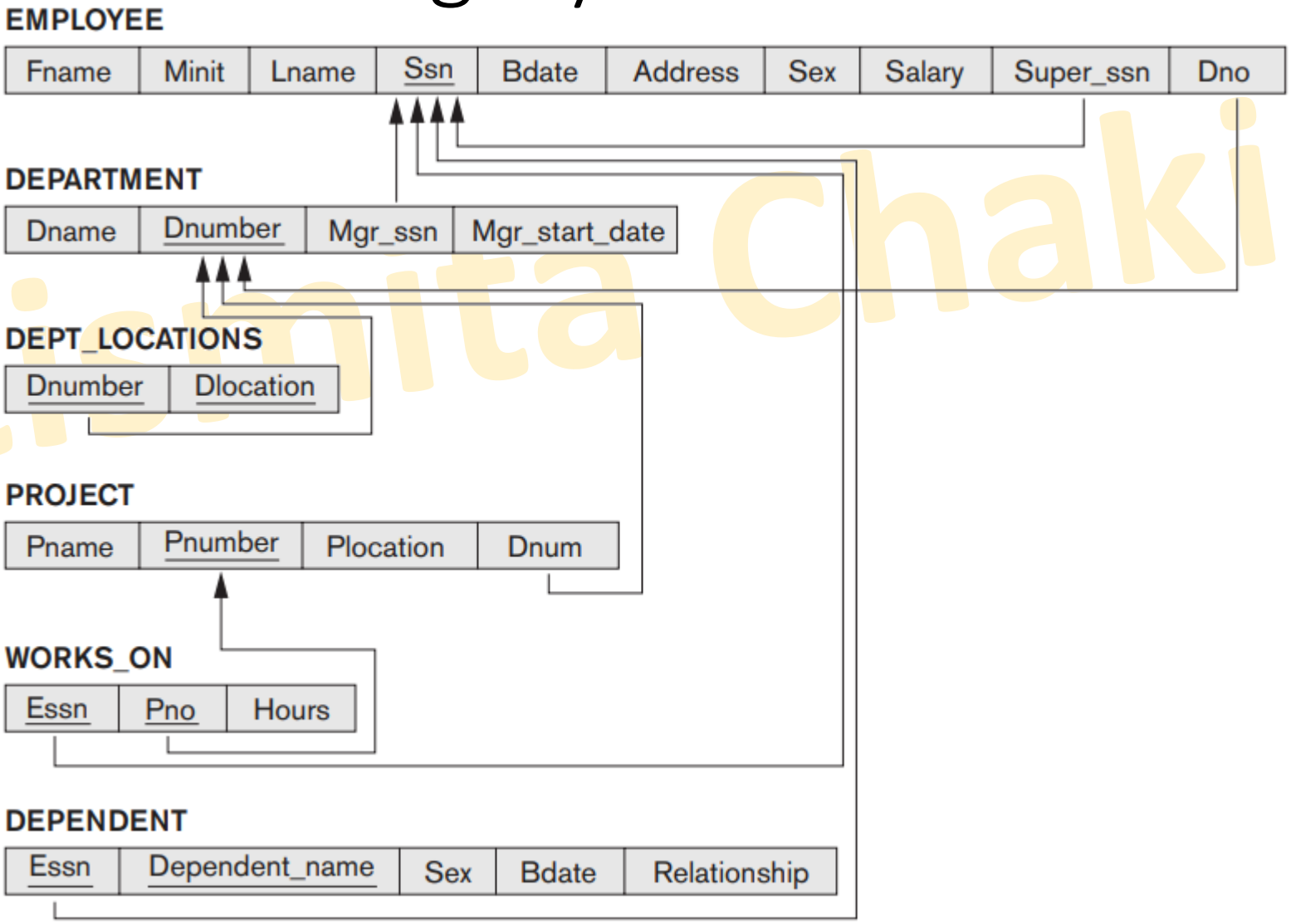
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

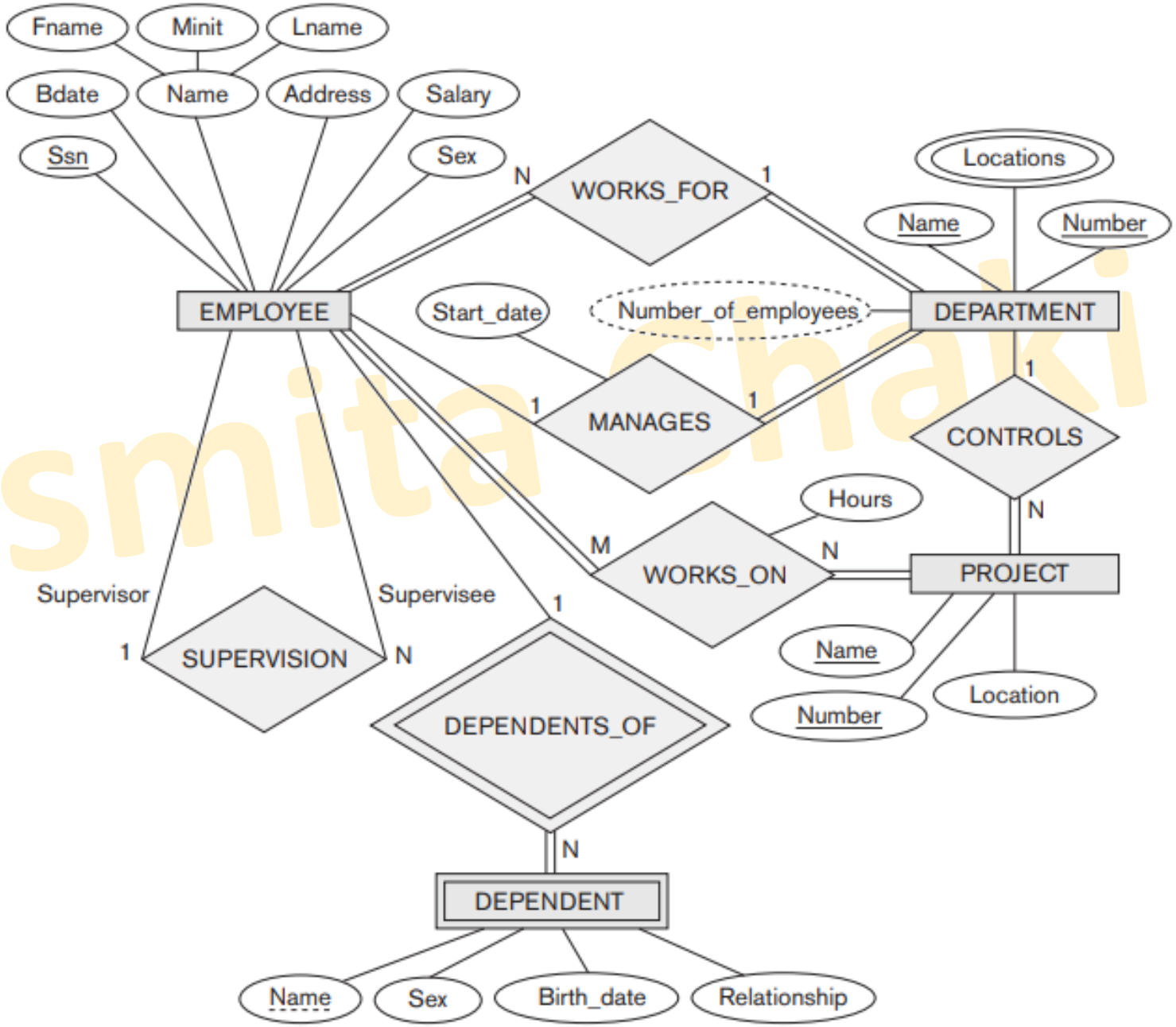
Relational Model Constraints: Integrity constraints: Referential integrity

- Referential integrity constraints displayed on the COMPANY relational database schema.



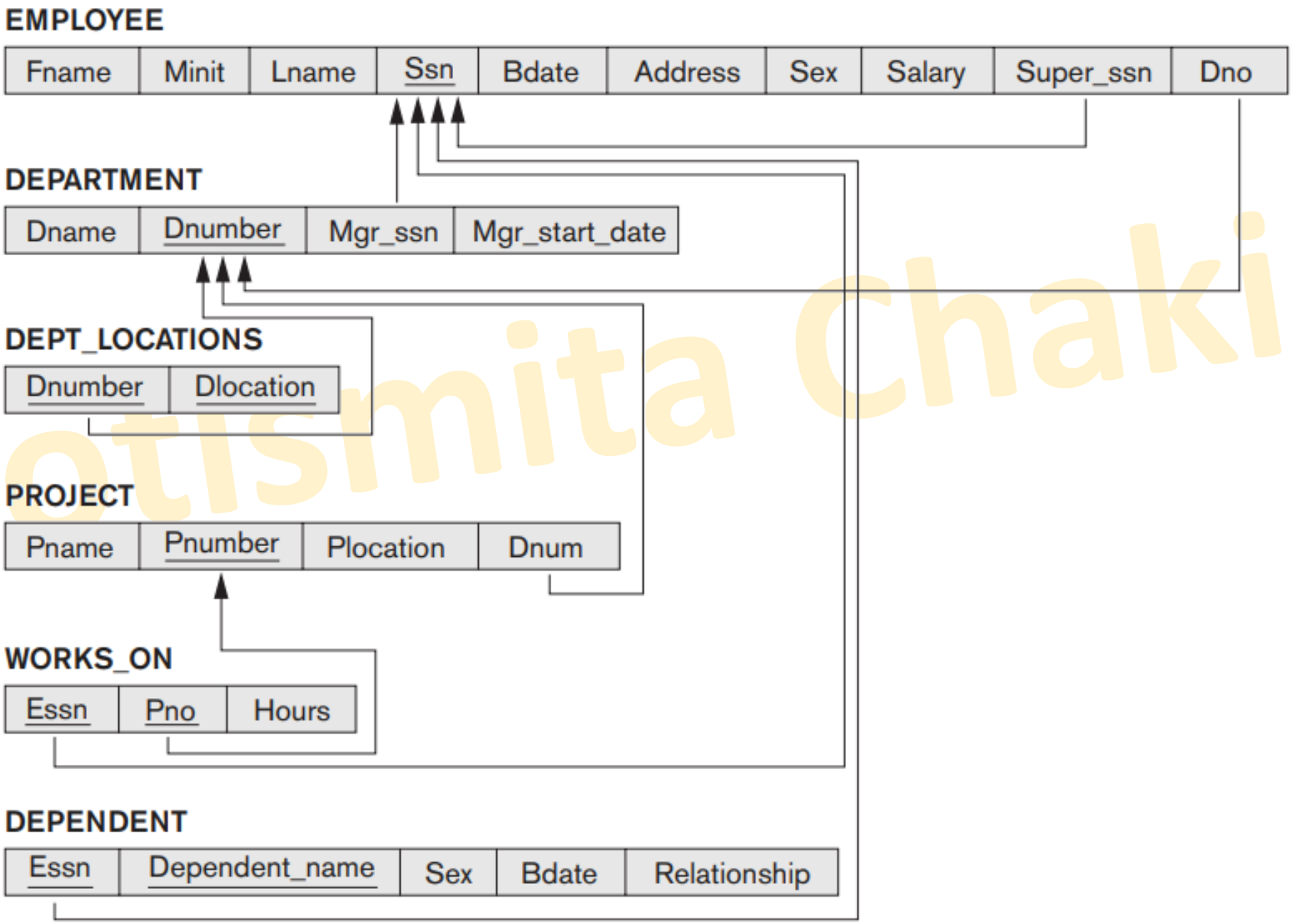
Mapping ER Model to Relational Schema

- The ER conceptual schema diagram for the COMPANY database



Mapping ER Model to Relational Schema

- Result of mapping the COMPANY ER schema into a relational database schema.



Mapping ER Model to Relational Schema

- Step 1: Mapping of Regular Entity Types.
  - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
  - Include only the simple component attributes of a composite attribute.
  - Choose one of the key attributes of E as the primary key for R.
- In our example, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT in relational database schema to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT from ER conceptual schema diagram .

Mapping ER Model to Relational Schema

- Step 1: Mapping of Regular Entity Types.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

Mapping ER Model to Relational Schema

- Step 2: Mapping of Weak Entity Types
  - For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
  - Include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W.
  - The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.
  - If there is a weak entity type E2 whose owner is also a weak entity type E1, then E1 should be mapped before E2 to determine its primary key first.

Mapping ER Model to Relational Schema

- Step 2: Mapping of Weak Entity Types
  - In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
  - We include the primary key Ssn of the EMPLOYEE relation—which corresponds to the owner entity type— as a foreign key attribute of DEPENDENT; we rename it Essn, although this is not necessary.
  - The primary key of the DEPENDENT relation is the combination {Essn, Dependent_name}, because Dependent_name (also renamed from Name) is the partial key of DEPENDENT.

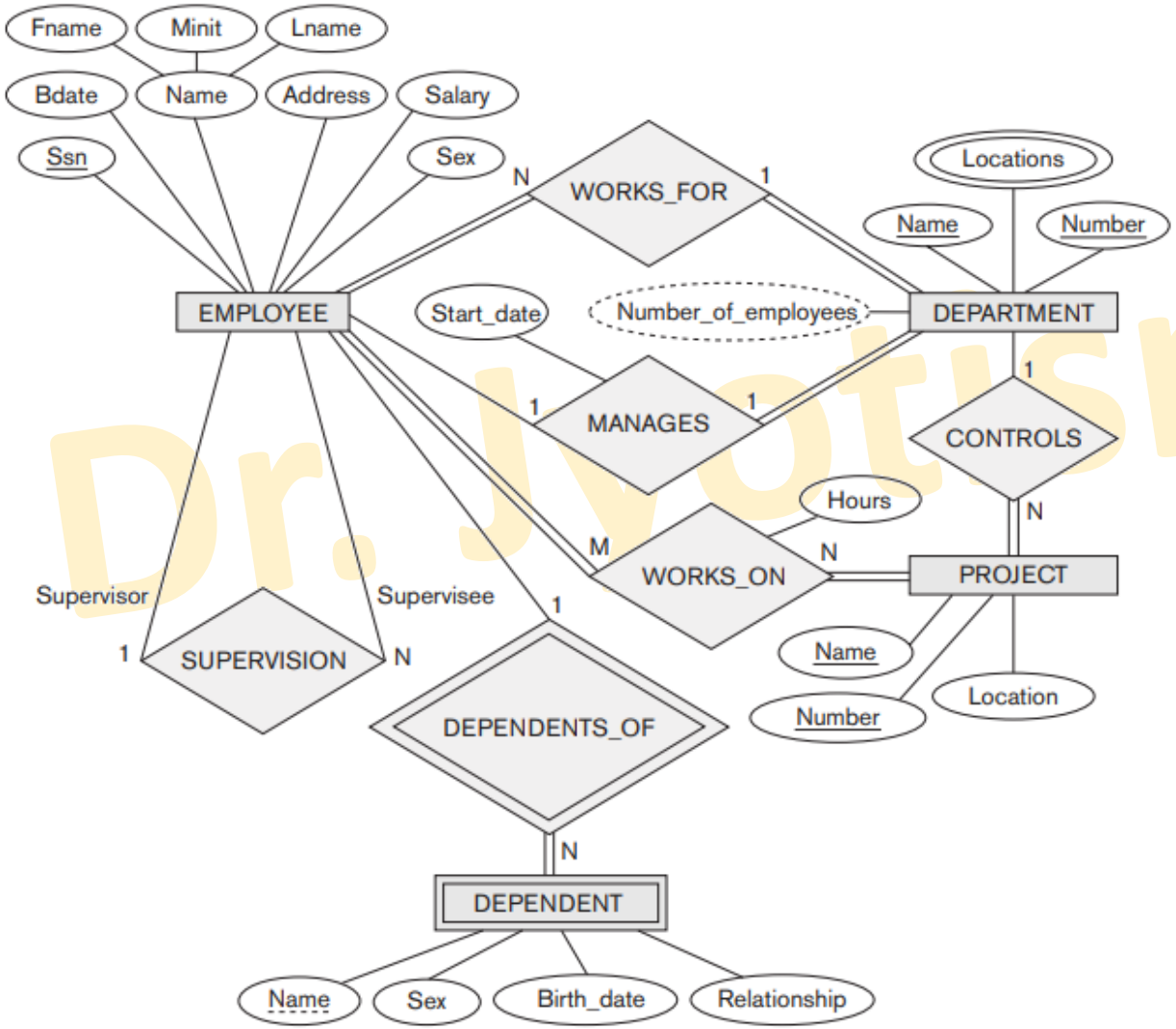
DEPENDENT

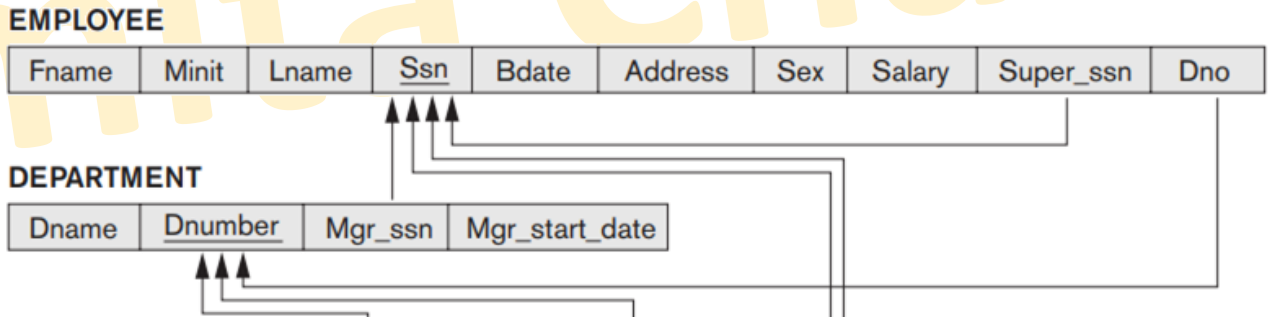
Essn	Dependent_name	Sex	Bdate	Relationship

Mapping ER Model to Relational Schema

- Step 3: Mapping of Binary 1:1 Relationship Types
  - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
  - Three possible approaches:
    1. the foreign key approach:
       - Choose one of the relations—S, say—and include as a foreign key in S the primary key of T.
       - It is better to choose an entity type with total participation in R in the role of S.
       - Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.
       - In our example, we map the 1:1 relationship type MANAGES from ER diagram by choosing the participating entity type DEPARTMENT to serve in the role because its participation in the MANAGES relationship type is total (every department has a manager).
       - We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it to Mgr_ssn.
       - We also include the simple attribute Start_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr_start_date

Mapping ER Model to Relational Schema





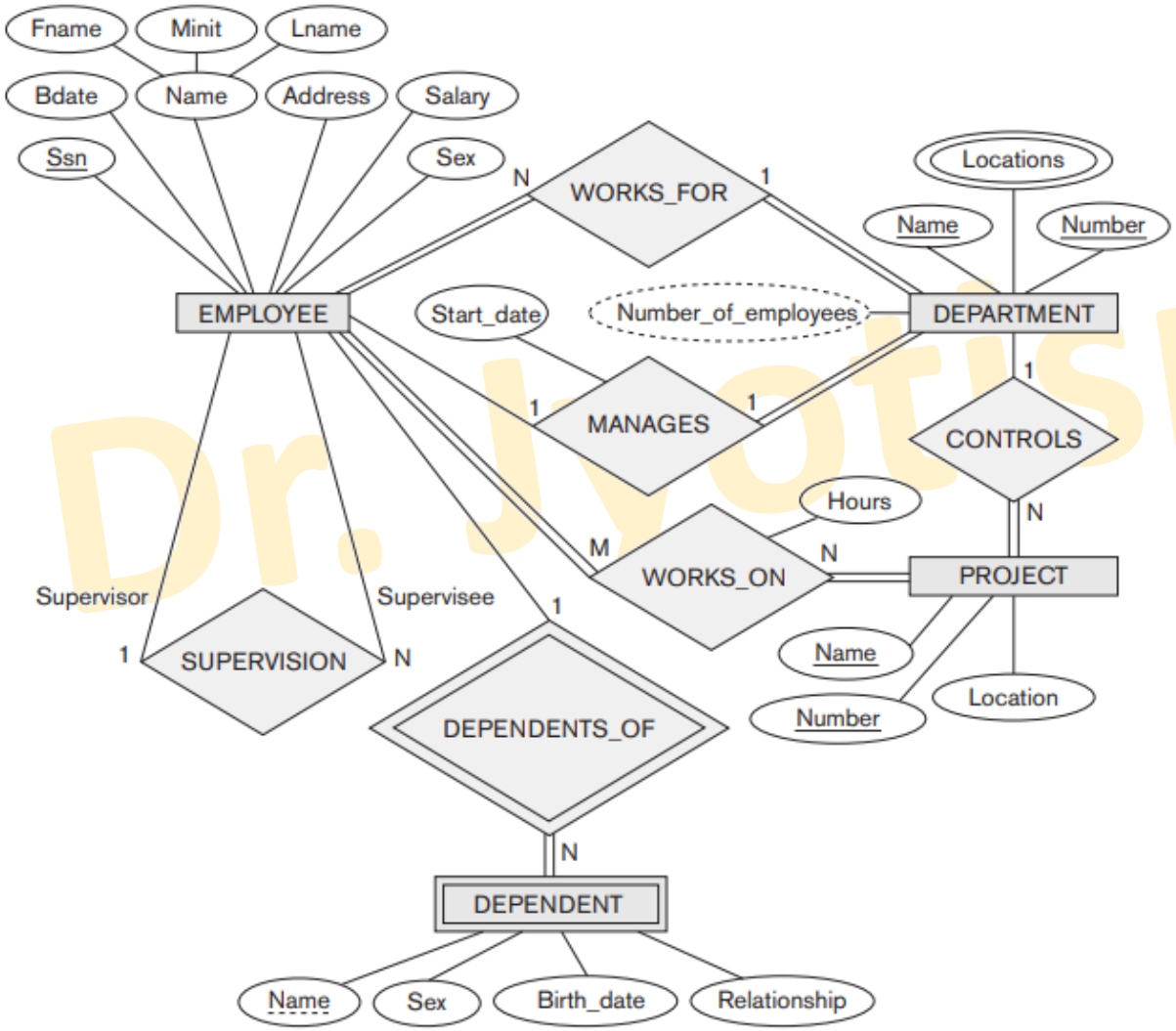
Mapping ER Model to Relational Schema

- Step 3: Mapping of Binary 1:1 Relationship Types
  - Three possible approaches:
    2. the merged relationship approach
       - An alternative mapping of a 1:1 relationship type is to merge the two entity types and the relationship into a single relation.
       - This is possible when both participations are total, as this would indicate that the two tables will have the exact same number of tuples at all times.
    3. the crossreference or relationship relation approach.
       - The third option is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.
       - This approach is required for binary M:N relationships.
       - The relation R is called a relationship relation (or sometimes a lookup table),
       - tuple in R represents a relationship instance that relates one tuple from S with one tuple from T.
       - The relation R will include the primary key attributes of S and T as foreign keys to S and T.
       - The primary key of R will be one of the two foreign keys, and the other foreign key will be a unique key of R.

Mapping ER Model to Relational Schema

- Step 4: Mapping of Binary 1:N Relationship Types
  - two possible approaches:
    1. The foreign key approach
       - For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type.
       - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R
       - Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S.
       - To apply this approach to our example, we map the 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION from the ER diagram.
       - For WORKS_FOR we include the primary key Dnumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it Dno.
       - For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super_ssn.
       - The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation.

Mapping ER Model to Relational Schema



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Mapping ER Model to Relational Schema

- Step 4: Mapping of Binary 1:N Relationship Types
  - two possible approaches:
    2. The relationship relation approach:
       - An alternative approach is to use the relationship relation (cross-reference) option as in the third option for binary 1:1 relationships.
       - We create a separate relation R whose attributes are the primary keys of S and T, which will also be foreign keys to S and T.
       - The primary key of R is the same as the primary key of S.
       - This option can be used if few tuples in S participate in the relationship to avoid excessive NULL values in the foreign key.

Mapping ER Model to Relational Schema

- Step 5: Mapping of Binary M:N Relationship Types
  - In the traditional relational model with no multivalued attributes, the only option for M:N relationships is the relationship relation (cross-reference) option.
  - For each binary M:N relationship type R, create a new relation S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Mapping ER Model to Relational Schema

- Step 5: Mapping of Binary M:N Relationship Types
  - In our example, we map the M:N relationship type WORKS_ON from the ER diagram by creating the relation WORKS_ON.
  - We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON and rename them Pno and Essn, respectively (renaming is not required; it is a design choice).
  - We also include an attribute Hours in WORKS_ON to represent the Hours attribute of the relationship type.
  - The primary key of the WORKS_ON relation is the combination of the foreign key attributes {Essn, Pno}.

WORKS_ON

Essn	Pno	Hours

Mapping ER Model to Relational Schema

- Step 6: Mapping of Multivalued Attributes
  - For each multivalued attribute A, create a new relation R.
  - This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute.
  - The primary key of R is the combination of A and K.
  - If the multivalued attribute is composite, we include its simple components.

Mapping ER Model to Relational Schema

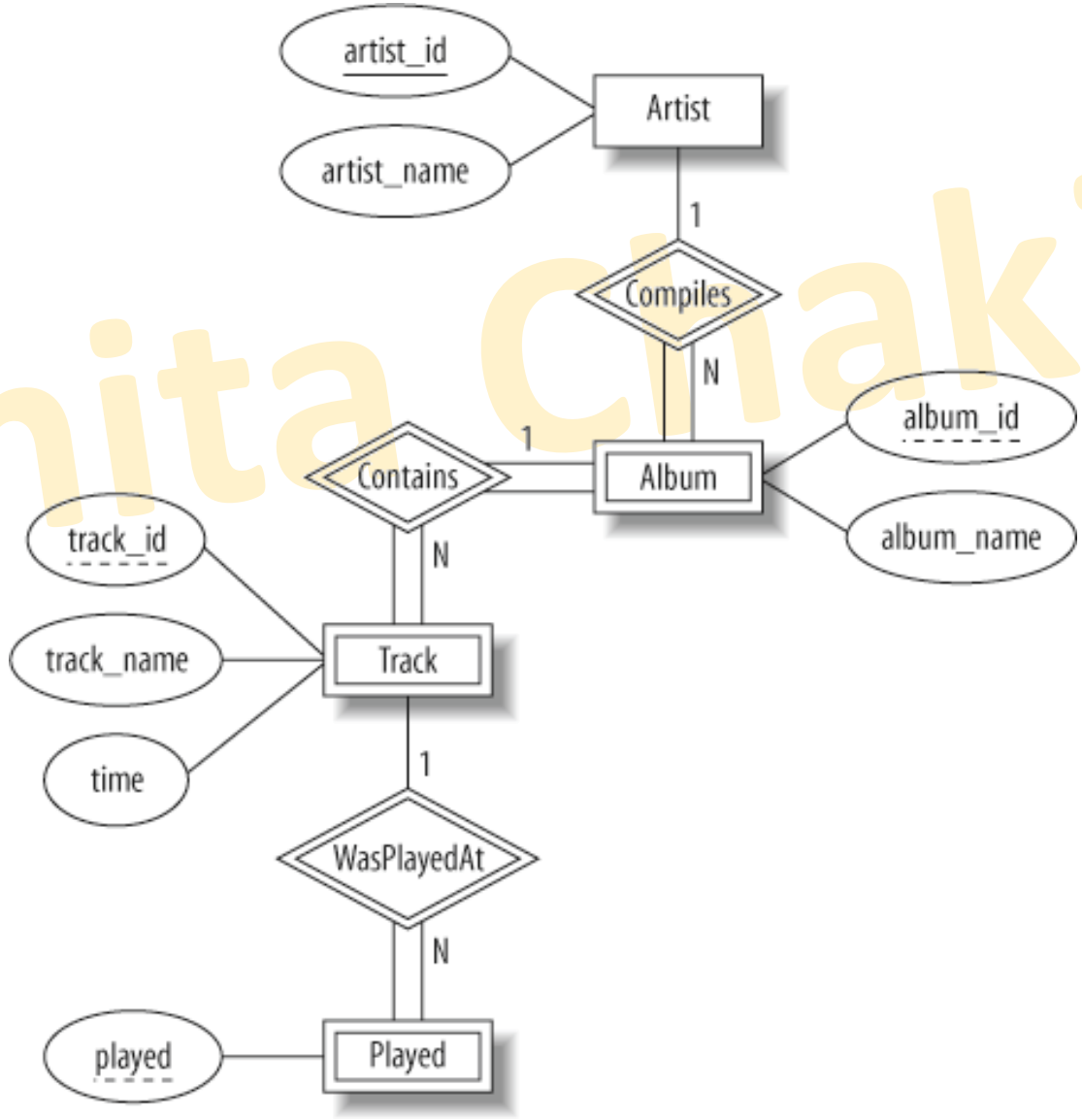
- Step 6
  - In our example, we create a relation DEPT_LOCATIONS.
  - The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, whereas Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation.
  - The primary key of DEPT_LOCATIONS is the combination of {Dnumber, Dlocation}.
  - A separate tuple will exist in DEPT_LOCATIONS for each location that a department has.

DEPT_LOCATIONS

Dnumber	Dlocation

ER Model examples: Music database

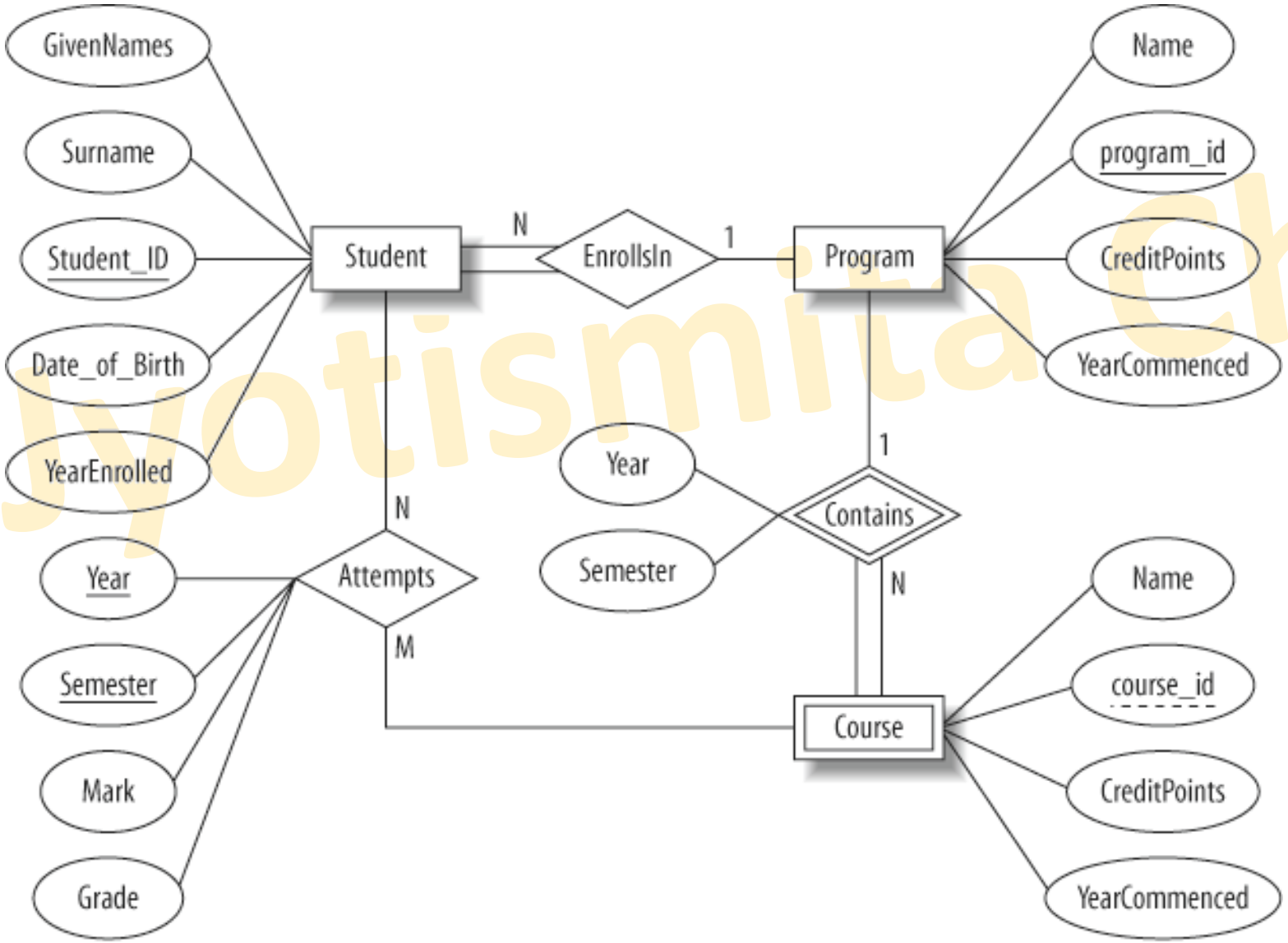
- Requirements for the database:
  - The collection consists of albums.
  - An album is made by exactly one artist.
  - An artist makes one or more albums.
  - An album contains one or more tracks
  - Artists, albums, and tracks each have a name.
  - Each track is on exactly one album.
  - Each track has a time length, measured in seconds.
  - When a track is played, the date and time the playback began (to the nearest second) should be recorded; this is used for reporting when a track was last played, as well as the number of times music by an artist, from an album, or a track has been played.



ER Model examples: University database

- Consider the following requirements list:
  - The university offers one or more programs.
  - A program is made up of one or more courses.
  - A student must enroll in a program.
  - A student takes the courses that are part of her program.
  - A program has a name, a program identifier, the total credit points required to graduate, and the year it commenced.
  - A course has a name, a course identifier, a credit point value, and the year it commenced.
  - Students have one or more given names, a surname, a student identifier, a date of birth, and the year they first enrolled. We can treat all given names as a single object—for example, “John Paul.”
  - When a student takes a course, the year and semester he attempted it are recorded. When he finishes the course, a grade (such as A or B) and a mark (such as 60 percent) are recorded.
  - Each course in a program is sequenced into a year (for example, year 1) and a semester (for example, semester 1).

ER Model examples: University database



ER Model examples: Flight database

- Consider the following requirements list:
  - The airline has one or more airplanes.
  - An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
  - An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
  - Each flight is carried out by a single airplane.
  - A passenger has given names, a surname, and a unique email address.
  - A passenger can book a seat on a flight.