Progettazione di una base di dati: Principi e Modello ER

Dall’uso all progettazione

- Finora abbiamo assunto di avere una base di dati relazionale (schema + istanze delle relazioni) e abbiamo visto come interrogarla (SQL)

- Adesso ci porremo un problema a monte di questo: come si fa a progettare una base di dati che:
  - garantisca un buon livello di qualità dei dati (anche nel lungo termine)
  - sia appropriata per le applicazioni che utilizzeranno i dati

La progettazione di una base di dati

- Ci sono due tipi di attività principali:
  - Progettazione della base di dati
  - Progettazione delle applicazioni che utilizzano la base di dati
- Qui ci focalizzeremo sulla progettazione concettuale e logica della base di dati
- La progettazione di applicazioni invece riguarda le funzionalità e le interfacce per accedere ai dati ed è normalmente trattata come parte dell'*ingegneria del software* o della *programmazione web*

La progettazione di una base di dati - II

- Le fasi principali della progettazione sono:
  1. Analisi dei requisiti
  2. Progettazione concettuale
  3. Progettazione logica

a cui seguono (anche intrecciate) le fasi di:

  4. Raffinamento dello schema logico (normalizzazione)
  5. Progettazione del livello fisico
  6. Progettazione applicativa e sicurezza

- Per ora ci concentriamo sui primi 3 passi.

La progettazione di una base di dati - III

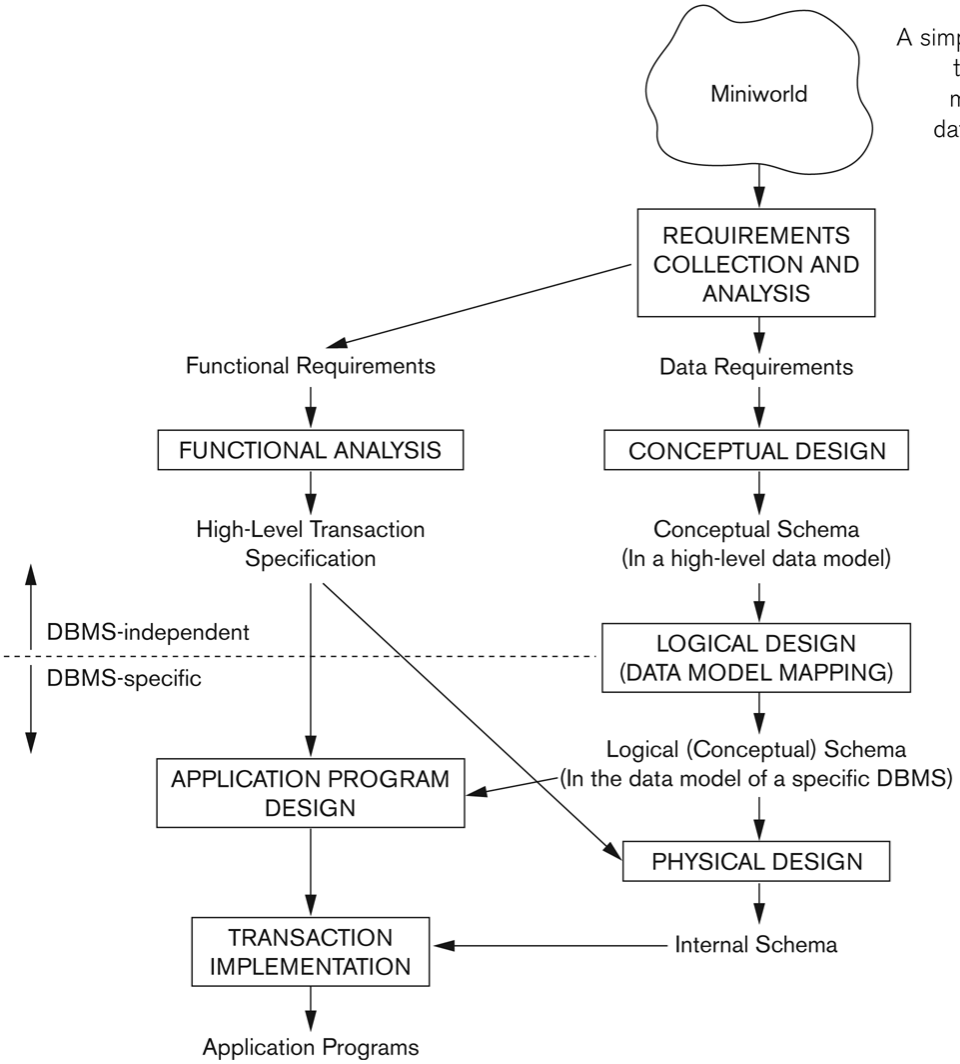


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Fase 1: analisi dei requisiti dei dati

Per le fasi 1-3 della progettazione di una base di dati dobbiamo rispondere alla seguente domanda:

- Quali dati dovranno essere raccolti nella BD?

Questo avviene mediante colloqui con il cliente, e con diversi gruppi di utenti; analizzando la documentazione disponibile; studiando le applicazioni che dovranno interagire con la BD

Per le fasi 4-6 invece dovremo successivamente considerare altre domande, tra le quali:

- Quali applicazioni dovranno usare questi dati?
- Quali saranno le operazioni più frequenti e quelle computazionalmente più costose?

Esempio: la base di dati “COMPANY”

- Immaginiamo che ci venga chiesto di progettare una base di dati per un’azienda «COMPANY»
- Ecco alcuni dei requisiti che sono emersi nei colloqui con il cliente :
  1. La società è organizzata in **DIPARTIMENTI**, ognuno dei quali ha un nome, un numero e un dipendente della società che ***gestisce*** il dipartimento. Un dipartimento può avere più sedi in luoghi diversi. Inoltre, dobbiamo tener traccia della data da cui un manager gestisce un dipartimento.
  2. Ogni dipartimento ***controlla*** un certo numero di **PROGETTI**. Ogni Progetto ha un nome unico, un codice unico ed è associato a un’unica sede.

Esempio

- Per ogni **DIPENDENTE**, la base di dati dovrà memorizzare il *social security number* (SSN), l'indirizzo, il salario, il sesso e la data di nascita
- Ogni dipendente ***lavora*** per un solo dipartimento, ma può lavorare contemporaneamente su diversi progetti. La BD dovrà tener traccia del numero di ore che ogni dipendente ***dedica*** a ogni progetto su base settimanale
- Inoltre, ogni dipendente ***ha*** un **supervisore** diretto, di cui la BD deve tener traccia.
- Ogni dipendente ***può avere*** un certo numero di **PERSONE A CARICO** (in Inglese: DEPENDENT).
  - Per ogni persona a carico, la base di dati dovrà registrare nome, sesso, data di nascita e tipo di relazione con il dipendente della società.

Il passo successivo

- Come faccio a costruire un modello che rappresenti in modo adeguato i requisiti espressi dal cliente?
  - Un modello è una rappresentazione semplificata di una porzione di mondo («mini-mondo»)
  - Il modello sarà *epistemologicamente* adeguato se permette di esprimere concretamente i fatti che sono conosciuti del mini-mondo di interesse (J. McCarthy & P. Hayes, 1969)
- Per le basi di dati, il modello più noto è usato a questo scopo è il modello *Entity-Relationship* (ER)

Il modello ER: concetti fondamentali

- Un'**entità** *(entity)*è un oggetto del mondo reale (o del nostro mini-mondo), distinguibile da altri oggetti, che vogliamo sia rappresentato nella BD
  - Per esempio, il DIPENDENTE John Smith, il DIPARTIMENTO di Ricerca e Sviluppo, il PROGETTO Prodotto_X

- A livello estensionale, uno specifico insieme di entità tra di loro simili (ovvero, che hanno attributi comuni) si chiama un **insieme di entità** *(entity set)*
  - Per esempio, l'insieme dei DIPENDENTI dell'Università di Trento, i suoi DIPARTIMENTI, i CORSI DI LAUREA

- Un **tipo di entità** (o *entity type*) è la definizione a livello intensionale delle entità a cui fanno riferimento diversi possibili *entity set*

Il modello ER: concetti fondamentali

- Gli **attributi** *(attributes)* sono le proprietà utilizzate per descrivere un'entità.
  - Per esempio, l'entità DIPENDENTE può avere attributi come *nome*, *SSN, indirizzo, sesso, data di nascita*

- Una specifica entità avrà uno specifico valore per ognuno dei suoi attributi.
  - Per esempio, un certo dipendente avrà Nome='John Smith', SSN='123456789', Indirizzo='731, Fondren, Houston, TX', Sesso='M', DataDiNascita='3 ottobre 1993'

- Ogni attributo è associato a un **insieme di valori** (*value set*) o tipo di dato (*data type*), chiamato **dominio.**
  - Per esempio, *string* per Nome, *integer* per SSN, …

Diversi tipi di attributo

Alcune versioni di ER ammettono diversi tipi di attributo:

- Semplice (*simple*)
  - Ogni entità ha un singolo valore atomico per quell'attributo. Per esempio '123456789' per SSN o 'M' per Sesso.
- Composto (*composite*)
  - L'attributo può avere molteplici componenti, per esempio:
    - Indirizzo: Scala#, Palazzina#, Via, Città, Stato, CAP, Paese
    - Nome: PrimoNome, SecondoNome, Cognome
  - Uno o più elementi di un attributo composto possono a loro volta essere composti.
- Multi-valore (*Multi-valued*)
  - Un'entità potrebbe avere molteplici valori per un certo attributo. Per esempio: CERTIFICAZIONI per un DIPENDENTE o TIPO_PATENTE per un AUTOMOBILISTA.

Noi utilizzeremo solo attributi semplici e discuteremo in seguito diversi modi di rappresentare gli attributi composti e multi-valore

Attributi chiave (*key attributes*)

- Ogni *entity type* ha un'attributo (o un insieme di attributi) che identifica in modo univoco ogni entità presente
- È detto **attributo chiave** dell'*entity type*.
- Esempi:
  - SSN per DIPENDENTE
  - NOME per un DIPARTIMENTO
  - CODICE per un PROGETTO

Attributi chiave - II

- Un attributo chiave può essere **composto**, ovvero essere dato dalla composizione di 2 o più attributi
  - Esempio: per un'AZIENDA, la chiave potrebbe essere RAGIONE_SOCIALE e PROVINCIA
- Un *entity type* può avere **più di una chiave** (in questo caso di parla di **chiavi candidate**):
  - Esempio: un'AUTOMOBILE potrebbe avere 2 chiavi candidate:
    - *NumeroTelaio* (anche conosciuto come VIN)
    - *Targa*

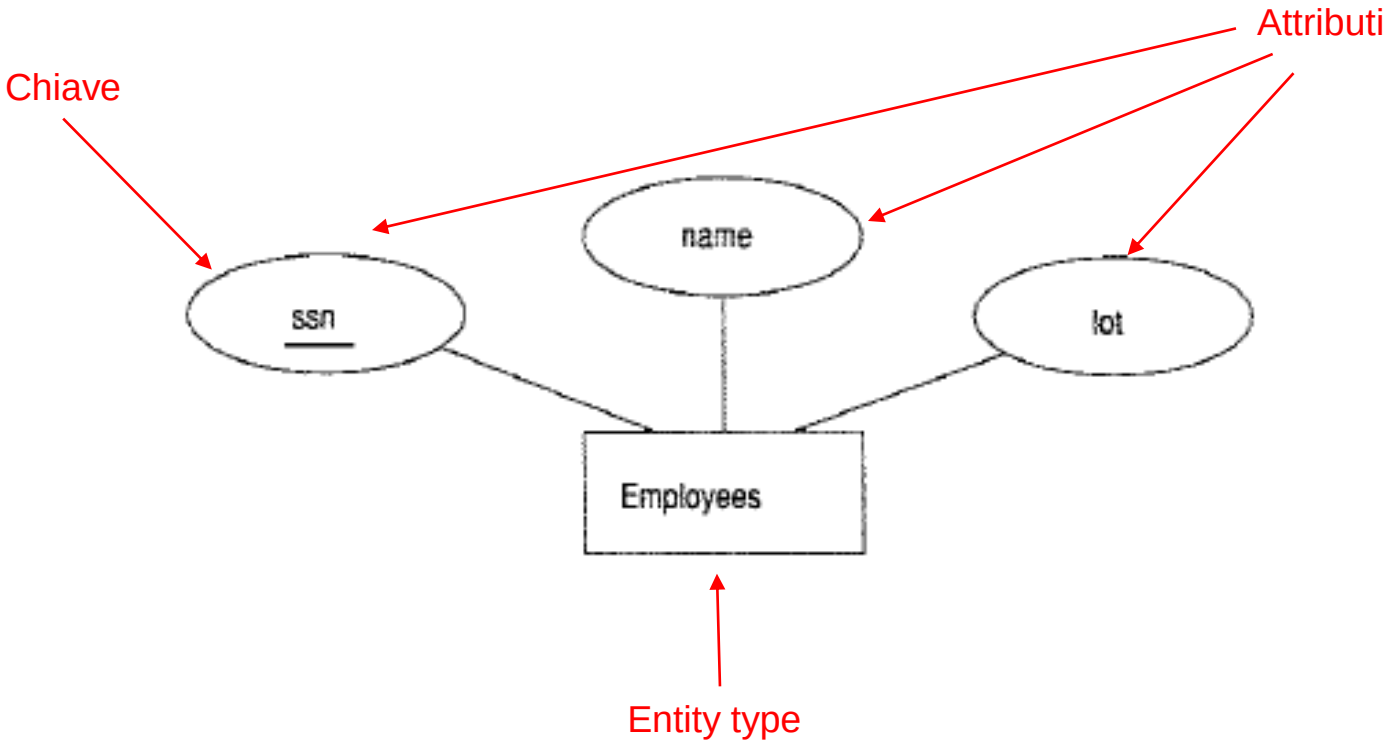
Attributi chiave - III

- Se un *entity type* ha più di una chiave candidata, una di esse viene scelta come **chiave primaria**:
  - Esempio: per AUTOMOBILE scegliamo come chiave primaria il *NumeroTelaio* (VIN)

- In ER, la chiave primaria è sempre sottolineata

Rappresentazione *Entity type*

- Un *entity type* è rappresentato come un rettangolo con il nome del tipo di entità
- Gli attributi sono rappresentati come ovali:
  - Ogni attributo è collegato al suo *entity type*
  - Gli attributi chiave sono sottolineati



Primo raffinamento: le **relazioni (relationships)**

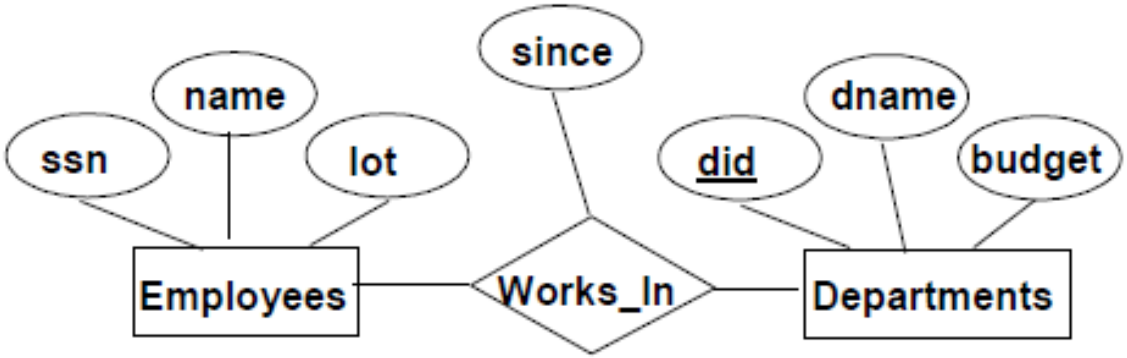
- La progettazione iniziale di solito non è completa, in particolare non tiene conto dell'esistenza di **relazioni tra tipi di entità**

- Una relazione (*relationship*) è un'associazione tra due o più entità
  - Per esempio, la relazione tra un IMPIEGATO e il DIPARTIMENTO per cui *lavora*

Relazioni e *Relationship Types* – Grado della relazione

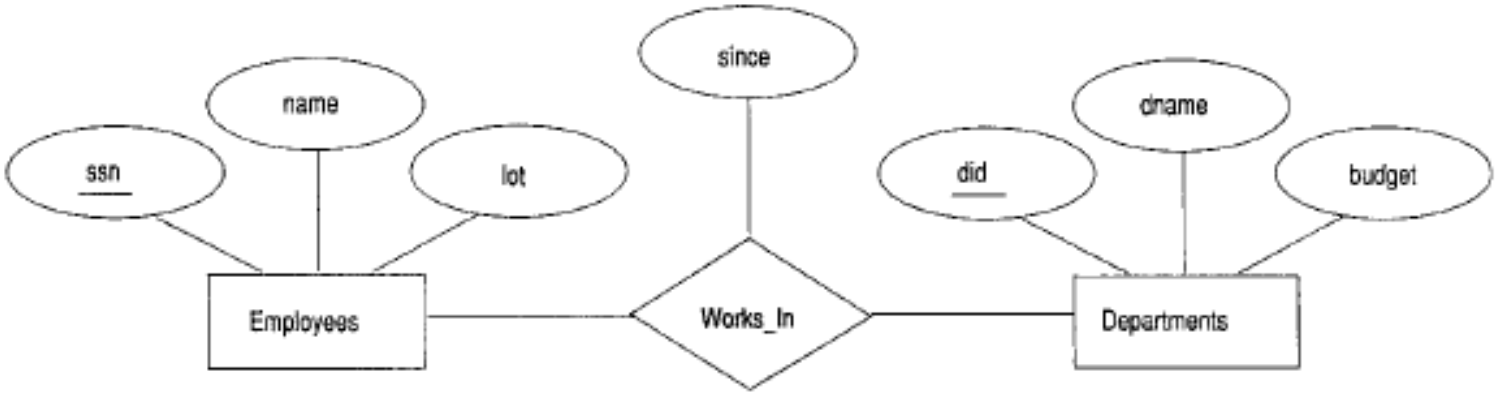
- Relazioni dello stesso tipo vengono raggruppate in un **relationship type**.
  - Per esempio, LAVORA_PER che lega impiegati (EMPLOYEES) e dipartimenti aziendali (DEPARTMENTS)

- Il numero di *entity types* che partecipano a un *relationship type* si dice il **grado** della relazione
  - Per esempio, LAVORA_PER è una relazione di grado 2 (binaria)



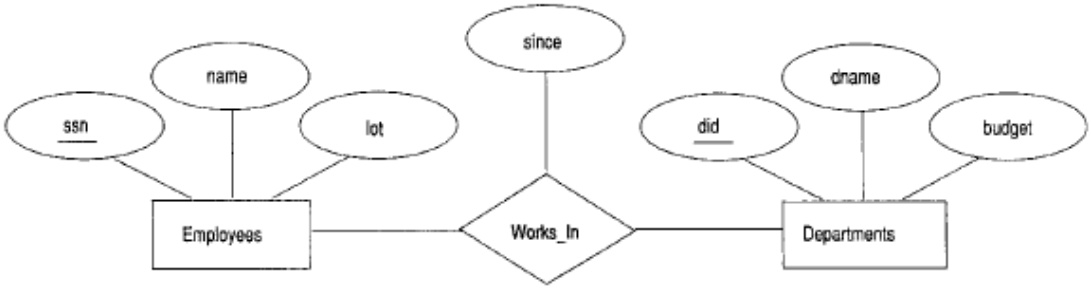
Relazioni e *Relationship Types* – Attributi della relazione

- Una relazione può avere uno o più attributi descrittivi
  - Per esempio, la relazione LAVORA_PER può essere qualificata da una data di inizio della collaborazione (l'attributo descrittivo *since*)
- Gli attributi descrittivi di una relazione forniscono informazioni sulla relazione tra le entità e **non** sulle entità che partecipano alla relazione



Relazioni e *Relationship Types* – Attributi della relazione

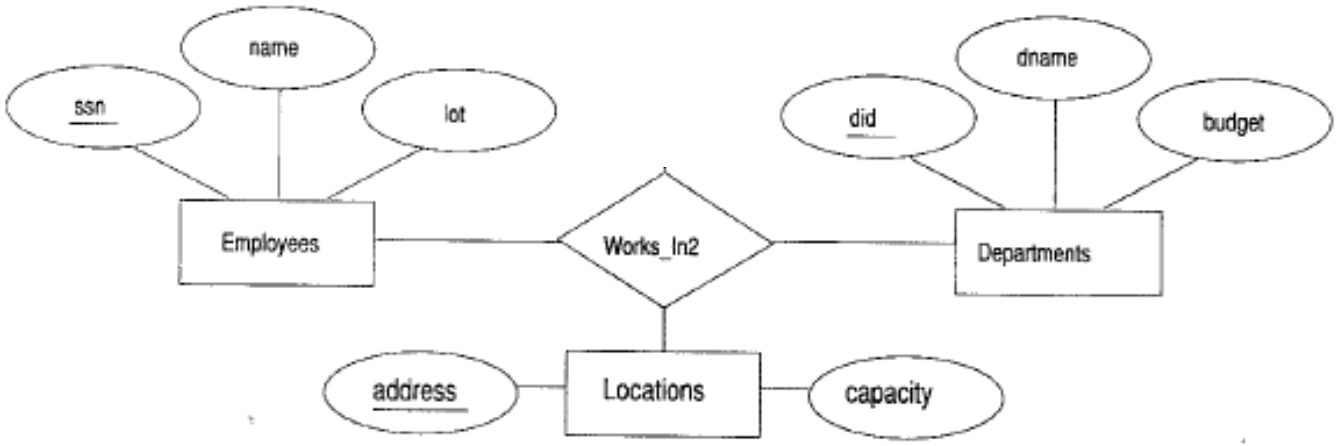
- NOTA BENE: questo significa che una relazione è identificata unicamente dalle entità che vi partecipano, e non dagli attributi descrittivi
  - Nell'esempio, ssn e did e non dal valore di *since*
- Questo implica che per ogni coppia <ssn,did> della relazione WORKS_IN non possiamo avere più di un valore per l'attributo *since* !!



  - E se fosse richiesto di registrare diversi periodi di lavoro di un impiegato in un certo dipartimento?

Relazioni e *Relationship Types* – Esempio di relazione di grado 3 (ternaria)

- Si assuma che un impiegato lavori per un dipartimento che può avere diverse sedi
- Questo richiede di rappresentare una relazione tra 3 diverse entità: impiegato, dipartimento e sede



In sintesi: Relationship type vs. Relationship set

- Relationship Type:
  - È lo schema che descrive una relazione
  - Identifica il nome della relazione e gli *entity types* che vi partecipano
  - Identifica anche i vincoli associati a una relazione
- Relationship Set:
  - È l'insieme di istanze della relazione che sono rappresentate nella base di dati in un certo istante
  - In un certo senso, è lo *stato attuale* di un *relationship type*

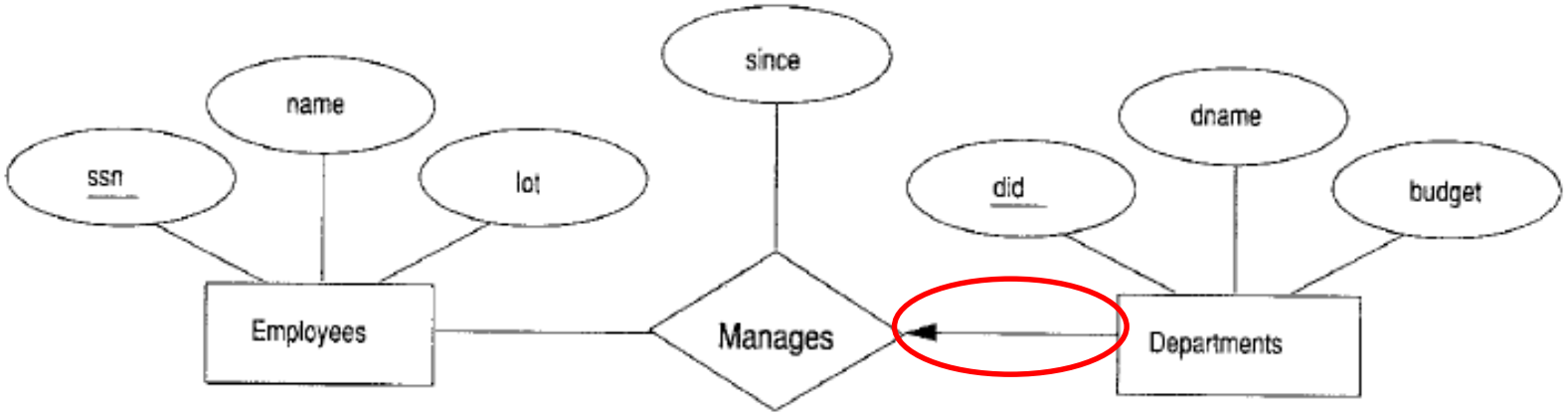
Vincoli sulle relazioni

- Nel seguito considereremo diversi tipi di vincoli su *Relationship Types*:
  - Vincoli di chiave (*key constraints*)
  - Vincoli di cardinalità (*cardinality ratio*)
  - Vincoli di esistenza o partecipazione (*participation constraints*)

- Ogni tipo di vincolo permette di catturare aspetti semantici specifici della relazione tra le entità partecipanti alla relazione

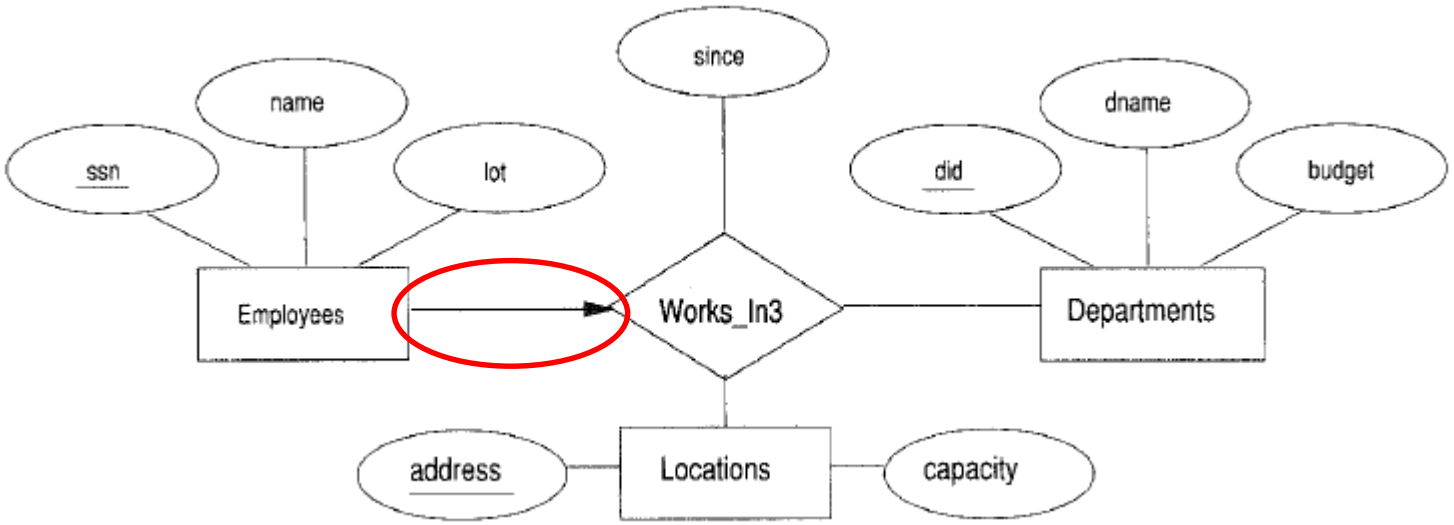
Vincolo di chiave (*key constraint*)

- Il vincolo di chiave permette di esprimere la condizione secondo cui un'entità può partecipare al massimo una volta in una relazione
  - Per esempio, la figura sotto illustra il vincolo in base al quale un Dipartimento può avere al massimo un manager (ma non viceversa!)



Vincolo di chiave per relazioni di grado > 2

- Il vincolo di chiave si generalizza a relazioni di grado >2 assumendo che una certa entità non possa comparire più di una volta in un *relationship set*
  - Per esempio, lo schema sotto esprime il vincolo che un impiegato non può lavorare in più di una sede di un dipartimento



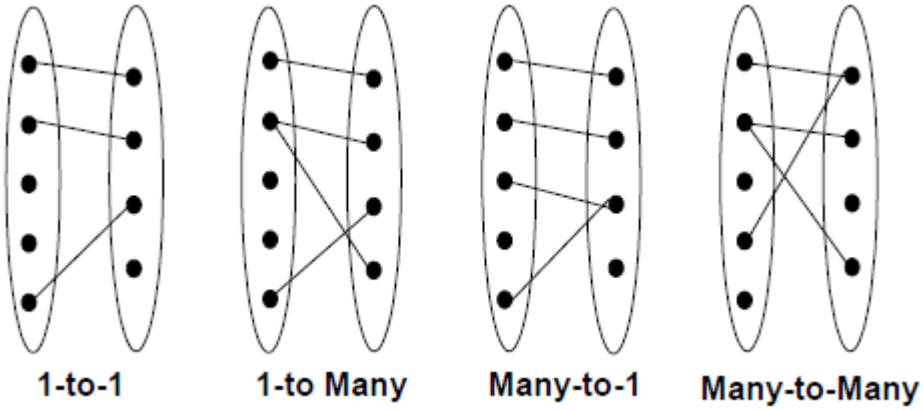
Vincoli di cardinalità

- Servono a specificare il numero massimo di volte in cui un'entità può COMPARIRE in un *relationship set*:
  - One-to-one (1:1)

  (es. PERSONA commissario_tecnico NAZIONALE)
  - One-to-many (1:N) or Many-to-one (N:1)

  (es. STUDENTE iscritto SCUOLA_DI_DOTTORATO)
  - Many-to-many (M:N)

  (es. STUDENTE frequenta CORSO)

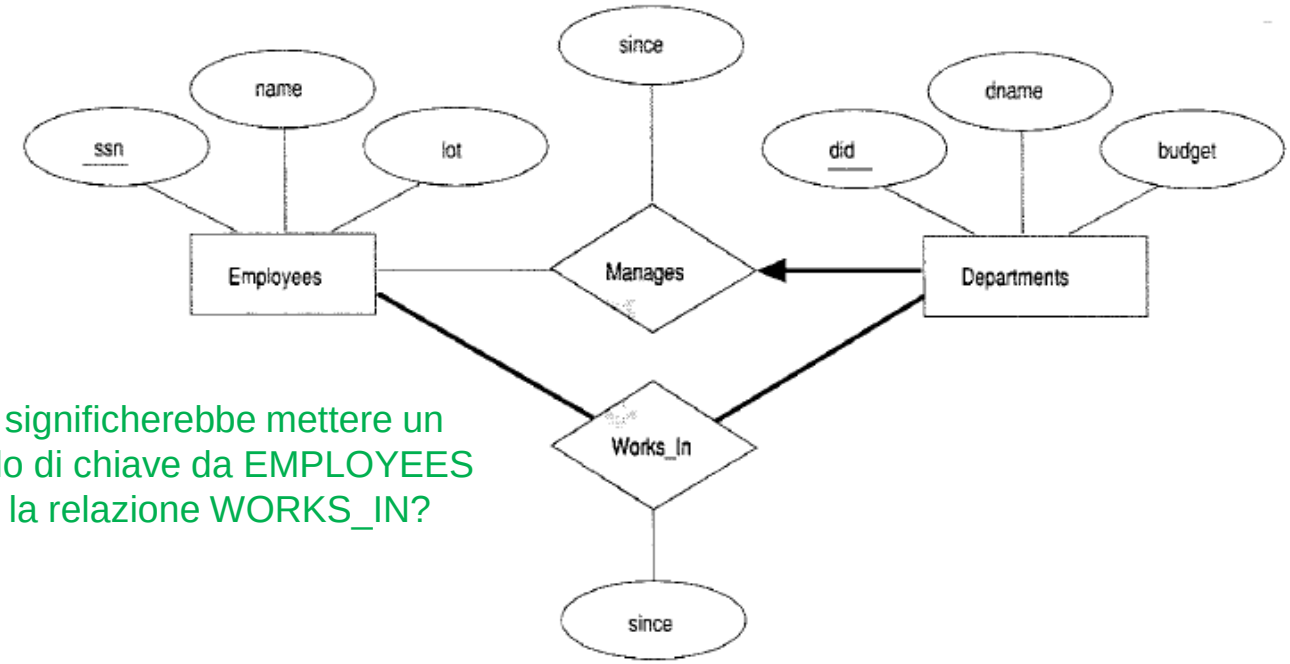


Vincolo di partecipazione (*participation constraint*)

- I vincoli di partecipazione specificano quante volte un'entità deve comparire **al minimo** in una relazione
  - Per questo sono detti anche **Existence Dependency Constraints**
- Un'entità può non comparire in un *relationship set* (**partecipazione parziale**)
  - → linea semplice in ER
- Un'entità deve comparire almeno una volta nel *relationship set* (**partecipazione totale**)
  - → linea in grassetto in ER

Vincolo di partecipazione - Esempio

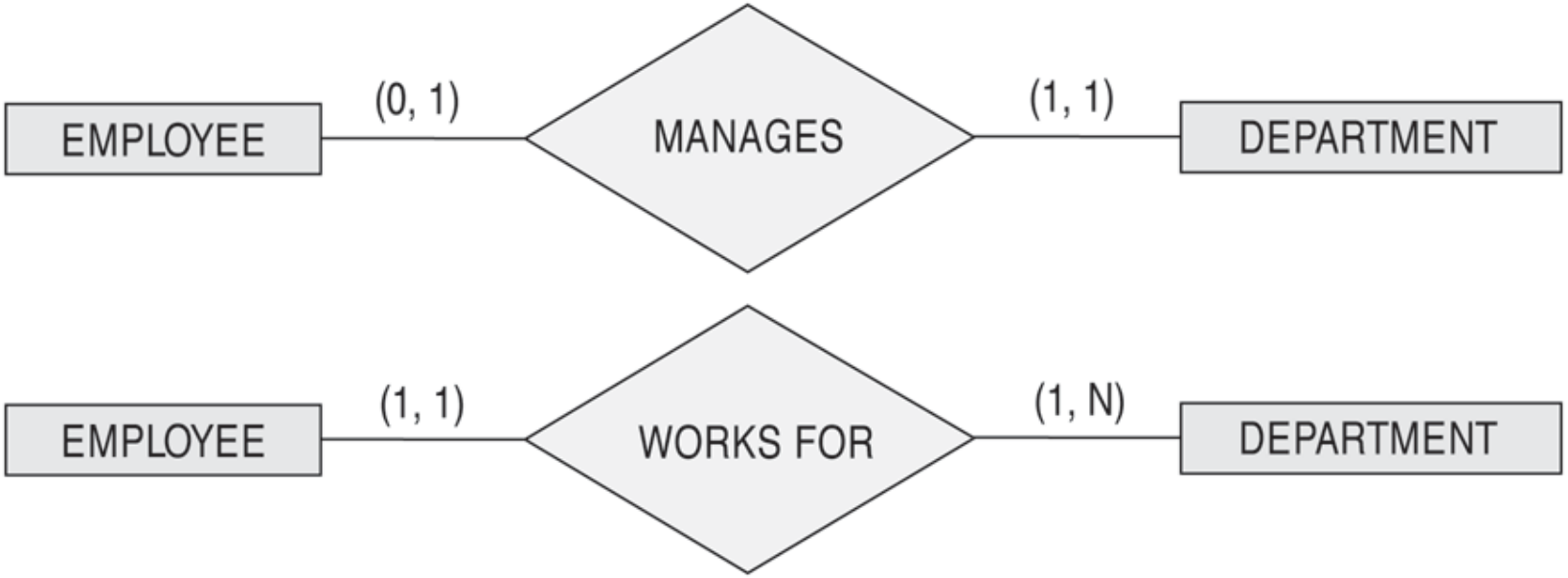
- DEPARTMENTS ha una partecipazione totale in MANAGES
  → Ogni dipartimento ha necessariamente un manager
- DEPARTMENTS e EMPLOYEES hanno una partecipazione totale in WORKS_IN (rappresentato dalla linea in grassetto)
  → Ogni impiegato deve afferire almeno a un dipartimento e ogni dipartimento deve avere almeno un impiegato che afferisce ad esso



Cosa significherebbe mettere un vincolo di chiave da EMPLOYEES verso la relazione WORKS_IN?

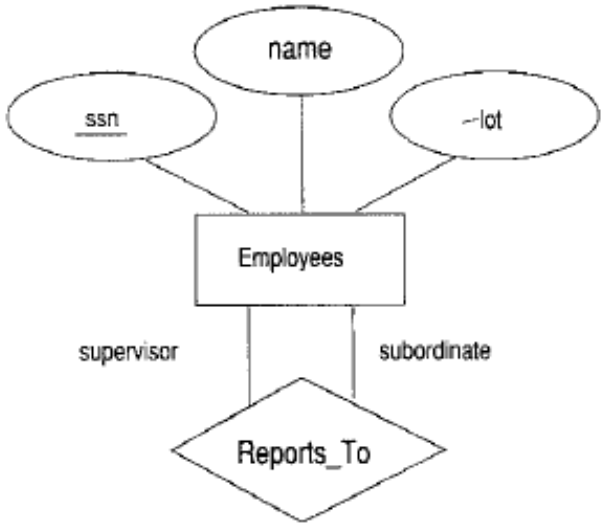
La notazione (min, max)

- I vincoli di cardinalità possono essere specificati anche indicando che ogni entità in E partecipa in almeno *min* e al massimo *max* istanze di R
- Default (assenza di vincoli): min=0, max=N (equivale a nessun vincolo) → non serve indicarlo
  - Per esempio:



Relazioni ricorsive

- Ci sono casi in cui una relazione vale tra entità dello stesso tipo ma in ruoli distinti
  - Per esempio: nella relazione REPORTS_TO, EMPLOYEE partecipa due volte ma con ruoli diversi (ed entità distinte):
    - Un'istanza di EMPLOYEE con il ruolo di Supervisore
    - Un'istanza di EMPLOYEE con il ruolo di Subordinato

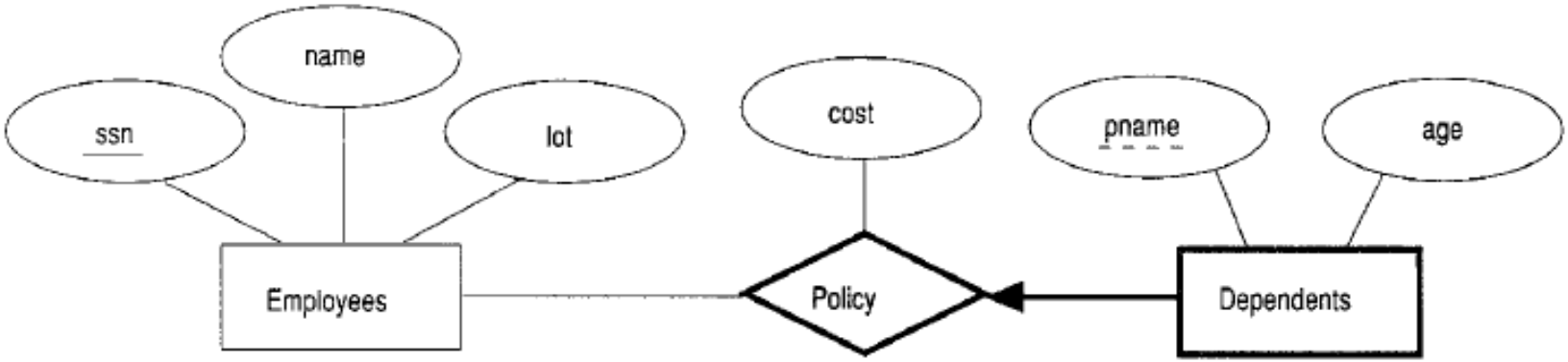


Entity Type deboli

- In alcuni casi possono esistere entità che non hanno un proprio attributo chiave e che quindi devono dipendere da altre entità per l'identificazione univoca. In questi casi si parla di **entità deboli**
- Un'entità debole deve partecipare a una **relazione identificante** a cui deve partecipare anche un'**entità identificante**.
- Le entità deboli sono quindi identificate da una combinazione di:
  - Una **chiave parziale** (che è propria dell'entità debole)
  - La **chiave dell'entità identificante** a cui è connessa mediante la relazione identificante

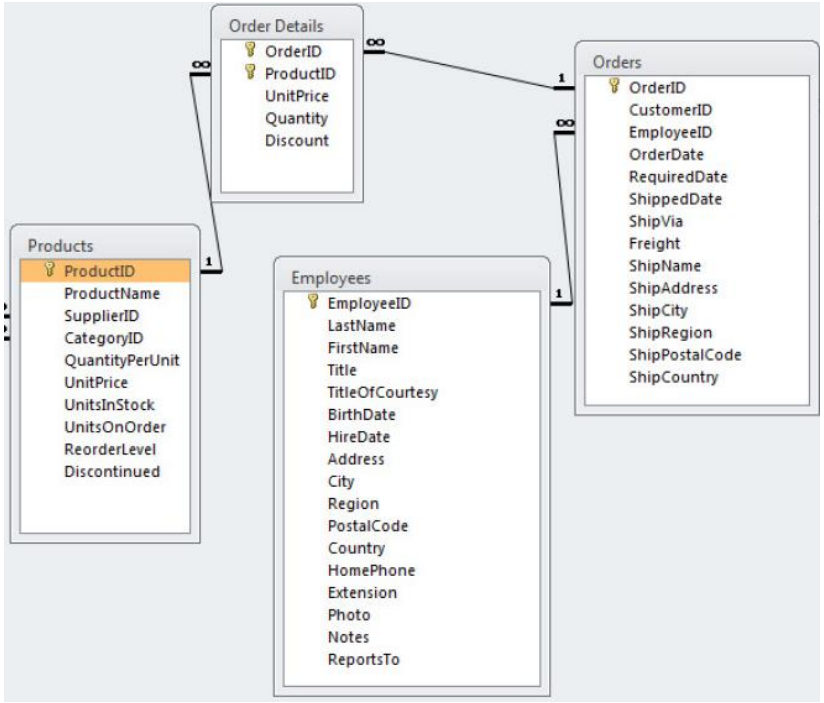
Entity Types deboli - Esempio

- Immaginiamo di voler registrare nella base di dati MyCOMPANY i familiari a carico dei nostri impiegati beneficiari di una polizza assicurativa aziendale:
  - Un'entità di tipo DEPENDENT (familiare a carico) è identificata dal nome del familiare e dallo specifico EMPLOYEE a cui è associato mediante la relazione identificante POLICY
  - Il nome del DEPENDENT è la *chiave parziale*
  - EMPLOYEE è l'entità identificante mediante l'associazione della relazione POLICY



Entity Type deboli - Esempio

- Immaginiamo di voler registrare gli ordini dell'azienda COMPANY, tenendo separati i dati generali di ogni ordine dai dati sui singoli prodotti (e loro quantità) inclusi dell'ordine
  - Un'entità di tipo ORDER (ordine), identificato dal suo ID (OrderID) corrisponde all'ordine
  - Un'entità di tipo ORDER_DETAILS contiene le singole righe dell'ordine (prodotto e quantità) ed è quindi identificata dall'OrderID a cui si riferisce e dall'ID del prodotto che compare nell'ordine (ProductID)
  - ORDER è l'entità identificante mediante l'associazione della relazione INCLUDE, mentre ORDER_DETAILS è l'entità debole.



Notazione per i vincoli sulle relazioni

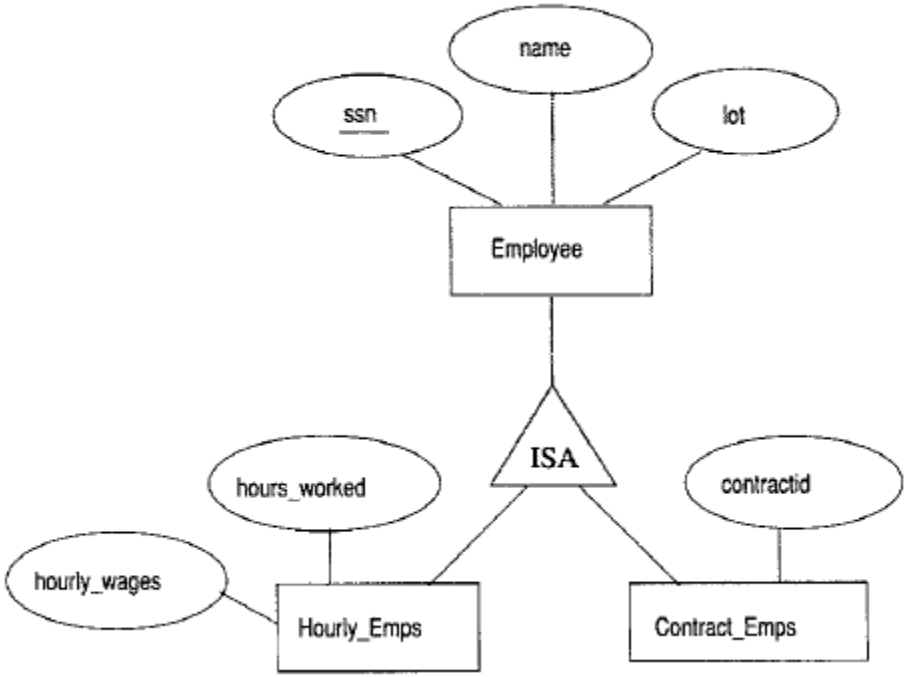
- Vincoli di chiave
  - Freccia semplice dall'entità vincolata alla relazione
- Vincoli di partecipazione
  - La partecipazione totale è rappresentata da una linea in grassetto, quella parziale da una linea semplice
- In alcuni casi è necessario utilizzare la notazione *(min, max)*
  - Esempio: quando *min* è maggiore di 1 o *max* è diverso da 1 o da N (es. 5)

Gerarchie di classi

- Talvolta è necessario organizzare Entity Types in gerarchie di classi
- Questo può avvenire per:
  - **Generalizzazione**: due Entity Type vengono generalizzati in un Entity Type che li ricomprende entrambi
  - **Specializzazione**: un Entity Type viene suddiviso in diversi sottoinsiemi, ognuno dei quali contiene entità che hanno delle caratteristiche in comune
- La specializzazione può avvenire secondo diverse dimensioni nello stesso modello, creando una **gerarchia multilivello**

Gerarchie di classi – Esempio

- Gli impiegati sono suddivisi tra dipendenti a contratto e dipendenti a ore
  - EMPLOYEE è specializzato in HOURLY_EMPS e CONTRACT_EMPS (ma potremmo voler aggiungere anche una specializzazione in impiegati junior e senior …)
  - HOURLY_EMPS e CONTRACT_EMPS sono generalizzati in EMPLOYEES



Perché introdurre gerarchie ISA

- Ci sono due motivazioni principali:
  - Per aggiungere attributi descrittivi che sono specifici di una certa sottoclasse
    - Esempio: il costo orario di un impiegato a ore
  - Per identificare il particolare sottoinsieme di entità che partecipano a una relazione
    - Esempio: solo i dipendenti a contratto potrebbero far parte di un circolo ricreativo aziendale

Vincoli sulle gerarchie ISA – Overlap

- **Overlap**: possono due classi avere entità in comune?
  - CONTRACT_EMPS e HOURLY_EMPS non possono avere sovrapposizioni
  - CONTRACT_EMPS e SENIOR_EMPS intuitivamente possono
- Nel secondo caso, scriviamo

  CONTRACT_EMPS **OVERLAPS** SENIOR_EMPS

- Il default è che NON ci sia overlap

Vincoli sulle gerarchie ISA – Coverage

- **Coverage** (completezza): le entità delle sottoclassi includono tutte le entità della sovraclasse?
  - TIGRI e PANTERE non «coprono» tutta la classe dei FELINI
  - JUNIOR_EMPS e SENIOR_EMPS presumibilmente «coprono» la classe degli EMPLOYEES
- Nel secondo caso, scriviamo:

  JUNIOR_EMPS **AND** SENIOR_EMPS **COVER** EMPLOYEES
- Il default è che il coverage NON valga

Sintesi su specializzazione / generalizzazione

- Possiamo avere 4 casi di specializzazione / generalizzazione:
  - No overlap, copertura totale
  - No overlap, copertura parziale
  - Overlap, copertura totale
  - Overlap, copertura parziale

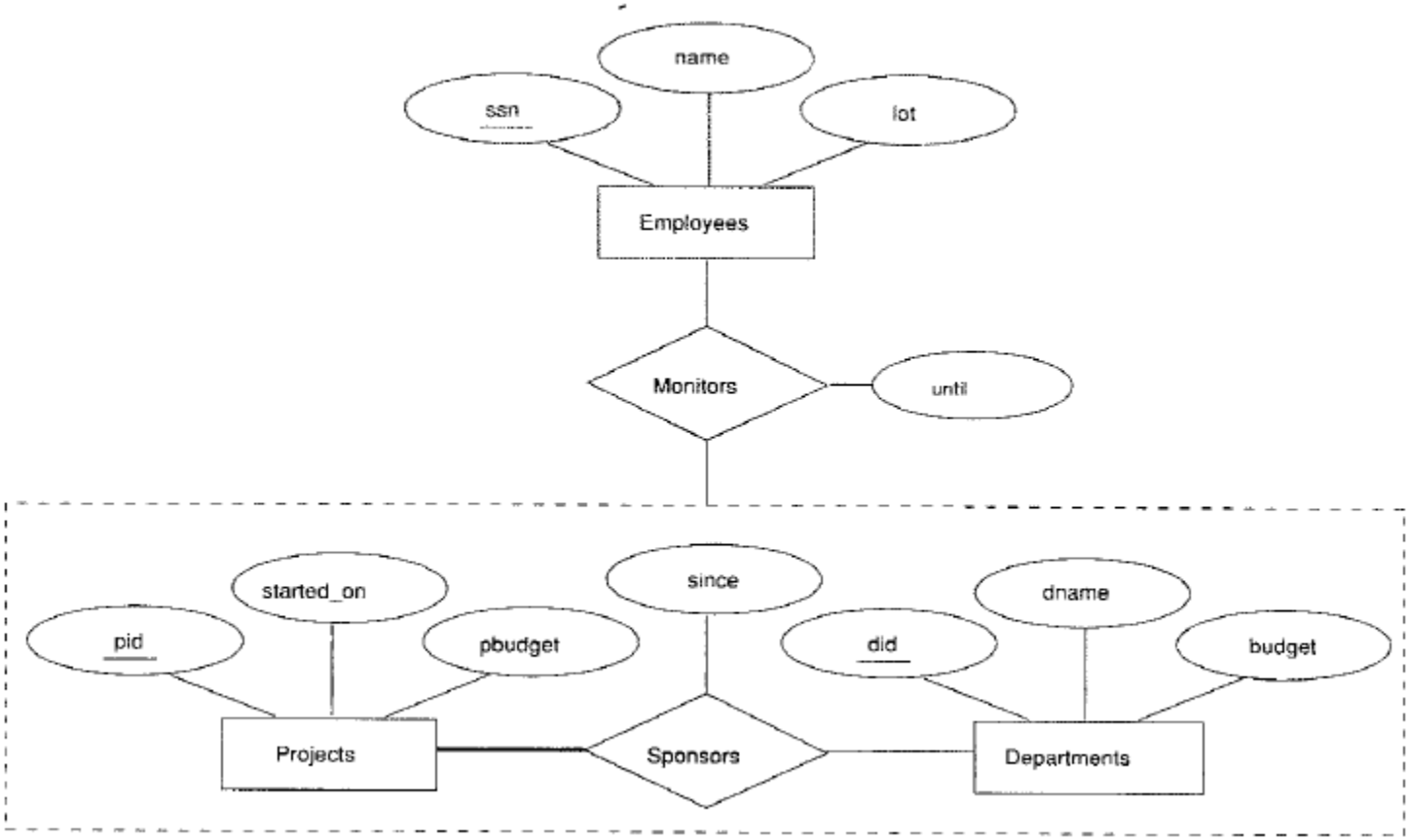
- Nota: la generalizzazione di solito (ma non necessariamente) è totale, dato che le superclassi sono derivate dalle sottoclassi

Aggregazione

- Finora abbiamo visto *relationship sets* che associano *entity sets*
- Può però capitare di voler associare un *entity set* con un *relationship set* che a sua volta associa 2 o più *entity set*
- Per questi casi, viene introdotto il concetto di **aggregazione**

Aggregazione - Esempio

- Immaginiamo che in COMPANY un impiegato possa essere incaricato di MONITORARE un PROGETTO che è SPONSORIZZATO da uno o più DIPARTIMENTI



Aggregazione vs. relazione ternaria

Avremmo potuto usare una relazione ternaria al posto dell'aggregazione per modellare l'esempio?

- L'uso di una relazione SPONSORS ternaria non ci avrebbe permesso di distinguere chiaramente l'esistenza di 2 relazioni (SPONSORS e MONITORS) con una semantica intuitivamente molto diversa
- L'uso di una relazione unica di grado 3 non ci avrebbe permesso di assegnare a MONITORS un suo attributo descrittivo specifico, molto diverso (e indipendente) dall'attributo SINCE della relazione SPONSORS
- Infine, l'uso di un'aggregazione rende molto più semplice e diretto l'uso di vincoli sulla relazione, come ad esempio il fatto che ogni sponsorizzazione possa essere monitorata al massimo da un impiegato (vincolo di chiave)

SCELTE DI PROGETTAZIONE CONCETTUALE CON ER

Questioni di modellazione in ER

- Definire un modello ER a partire dai requisiti raccolti dagli utenti non è un processo semplice né lineare
- Spesso lo stesso insieme di requisiti può essere modellato concettualmente in modi diversi
- Non tutti i vincoli di dominio possono essere rappresentati in ER per mancanza di espressività del modello

Scelte di modellazione in ER

- Qui discuteremo le scelte più frequenti che un progettista deve affrontare nello sviluppo di un modello ER, ovvero:
  - *Entità* vs. *Attributo*
  - *Entità* vs. *Relazione*
  - *Quali relazioni usare? Di che grado?*
  - *Relazioni* vs. *Aggregazioni*

- Per concludere, verranno evidenziati alcune delle più comuni limitazioni di espressività di ER

Entità o Attributo?

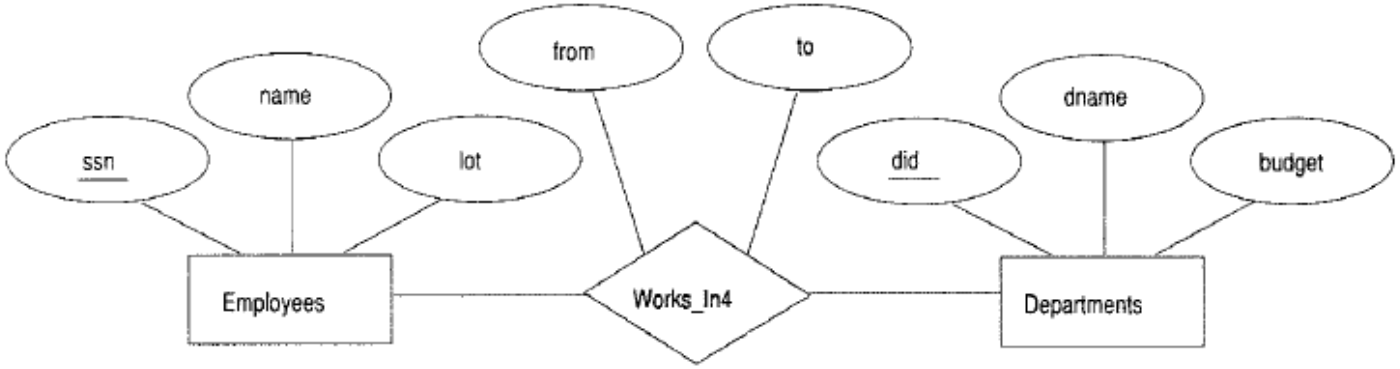
- In molti casi la scelta non è vincolante, ovvero entrambe le soluzioni possono andare bene (nel qual caso si tende a preferire la scelta dell'attributo per semplicità del modello)
- Tuttavia, ci sono due casi in cui è necessario creare l'*entity type*:
  1. Quando sappiamo che potremmo dover registrare più valori per ogni entità
  2. Quando vogliamo scomporre ulteriormente l'informazione in sotto-componenti

Entità o Attributo? Esempio

- Immaginiamo di voler associare ai dipendenti anche informazione sul loro indirizzo
- Lo facciamo aggiungendo un attributo o creando un'*entity type* INDIRIZZO?
- È necessario creare l'*entity type* quando:
  1. dobbiamo poter registrare più indirizzi per lo stesso dipendente
  2. vogliamo scomporre l'indirizzo in via, comune, provincia, stato, ecc. per permettere query specifiche (es. tutti i dipendenti che vivono nello stesso comune)

Entità o Attributo? Esempio 2

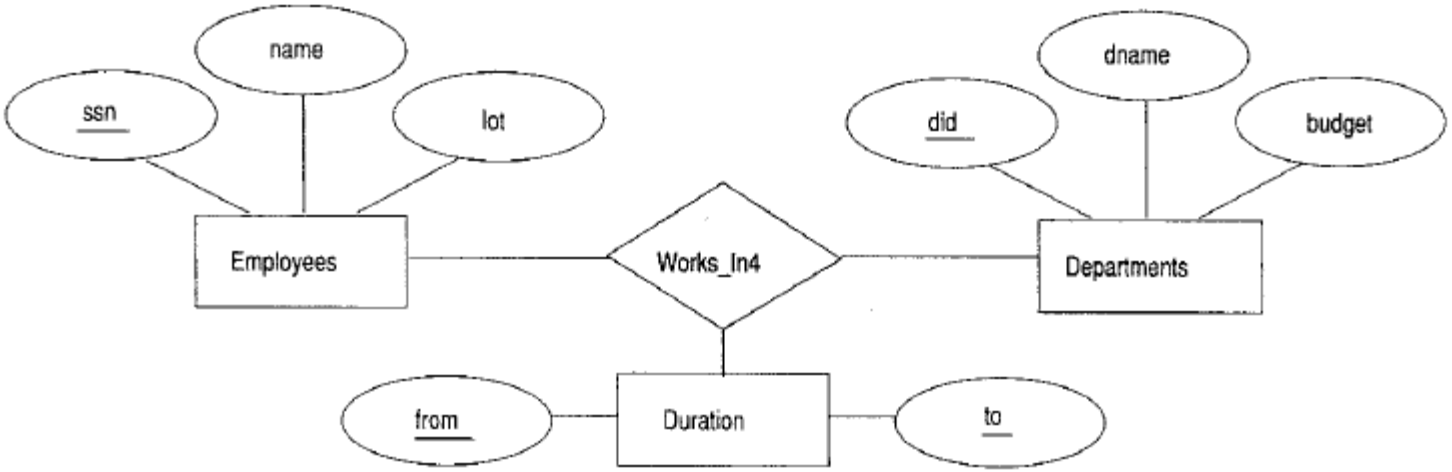
- Immaginiamo di voler associare alla relazione WORKS_IN anche le date di inizio e fine
- Opzione 1: attributi *from* e *to* associati a WORKS_IN



- Questo però ci consente di specificare **solo un periodo di lavoro** presso quel dipartimento per ogni impiegato!

Entità o Attributo? Esempio 2

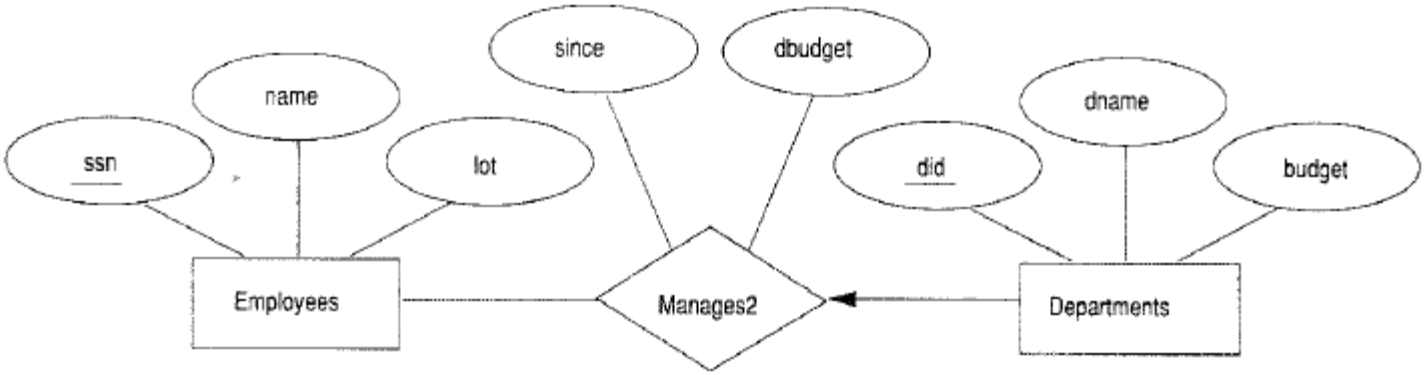
- Se vogliamo poter registrare più di un periodo di lavoro di un dipendente presso un dipartimento, dobbiamo necessariamente creare un nuovo ENTITY TYPE



- La scelta quindi dipende dai requisiti, che vanno esplicitati con molta attenzione!

Entità o Relazione?

- Immaginiamo che ogni impiegato che gestisce un dipartimento abbia anche un budget



- Ma cosa succede se un impiegato può essere manager di più dipartimenti con un budget complessivo (e non per singolo dipartimento)?

Entità o Relazione?

- Usare il modello con gli attributi associati alla relazione crea due problemi diversi ma di uguale importanza:
  - Da un lato, il valore complessivo del budget sarebbe riportato identico per tutti i dipartimenti diretti da quel dipendente (ridondanza → ne riparleremo nella normalizzazione)
  - Dall'altro, si passa il messaggio sbagliato che il budget è associato con la relazione, mentre è associato al manager

Entità o Relazione?

- Una possibile soluzione è quella di specializzare EMPLOYEES con una sotto-classe MANAGERS e assegnare alla sotto-classe gli attributi SINCE e DBUDGET
  - Questo ha il vantaggio di evidenziare che il budget è una proprietà di MANAGERS, non della loro relazione con un singolo dipartimento

- Se un manager può avere diverse date di inizio di incarico presso ogni DIPARTIMENTO che gestisce, un ulteriore raffinamento potrebbe riportare l'attributo SINCE sulla relazione (togliendolo dalla sotto-classe MANAGERS)

Relazioni binarie o ternarie?

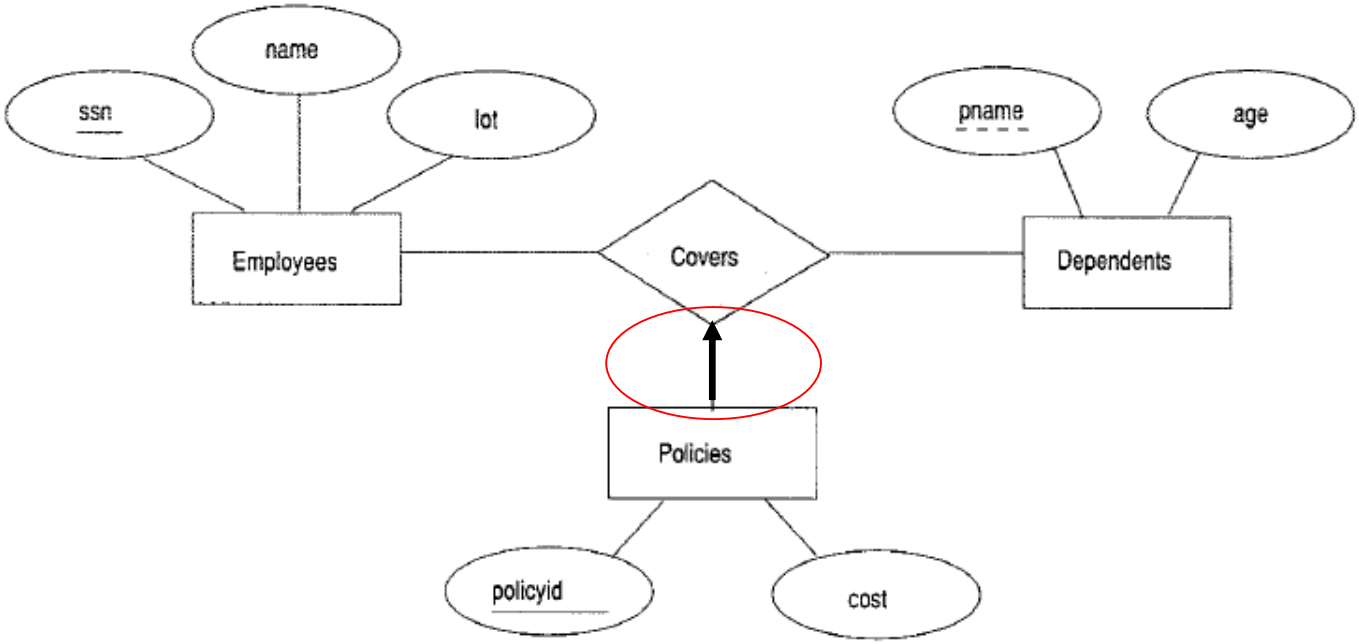
- In alcuni casi può presentarsi la decisione se modellare una relazione tra 3 *entity types* mediante una relazione di grado 3 o mediante 2 relazioni di grado 2
- Quando ci sono 3 entità coinvolte in un requisito, è fondamentale chiedersi se si tratta veramente di una relazione tra 3 entità o se invece non siamo di fronte a due relazioni distinte tra coppie di entità
- Vediamo un esempio del secondo caso

Relazioni binarie o ternarie: Esempio

- Riprendiamo l'esempio dei familiari a carico e aggiungiamo alcuni nuovi requisiti:
  - Ogni polizza deve essere posseduta da un impiegato
  - Una polizza non può essere posseduta congiuntamente da 2 o più impiegati
- Come è preferibile modellare questo scenario?

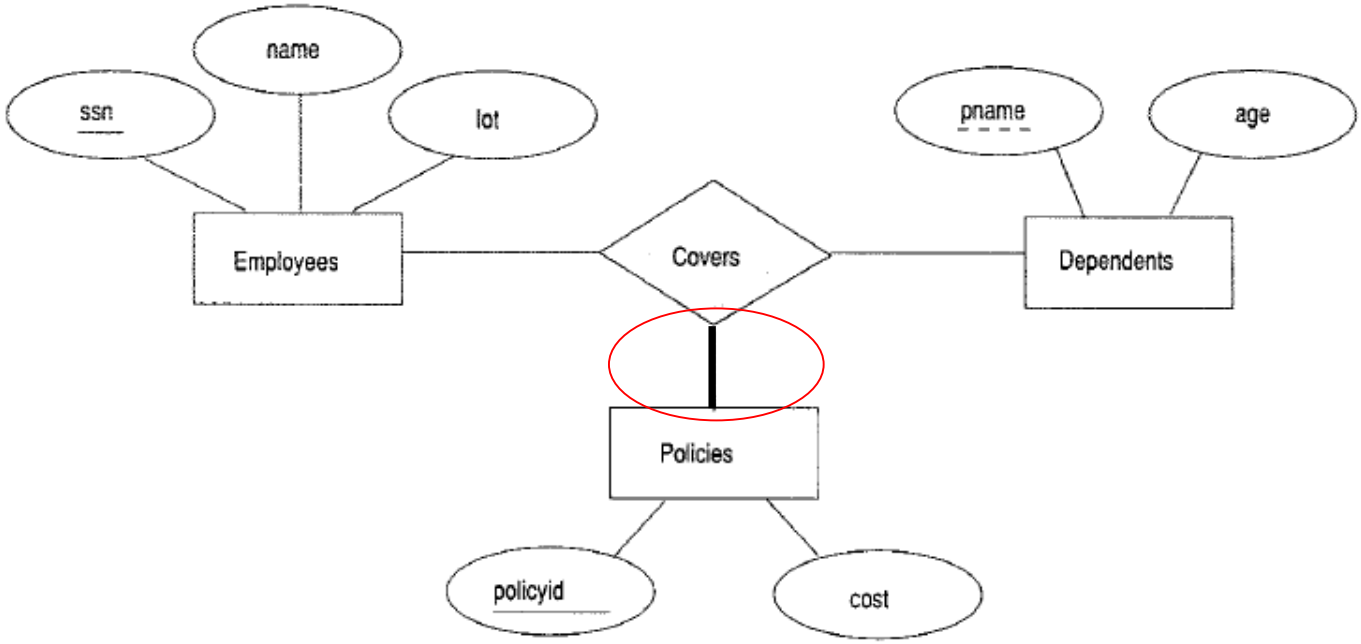
Ipotesi #1: relazione ternaria con vincoli

- Il primo requisito potrebbe essere implementato imponendo un vincolo di chiave da POLICIES a COVERS
  - Problema: questa soluzione ha l'effetto indesiderato di imporre che una polizza possa coprire solo un familiare a carico



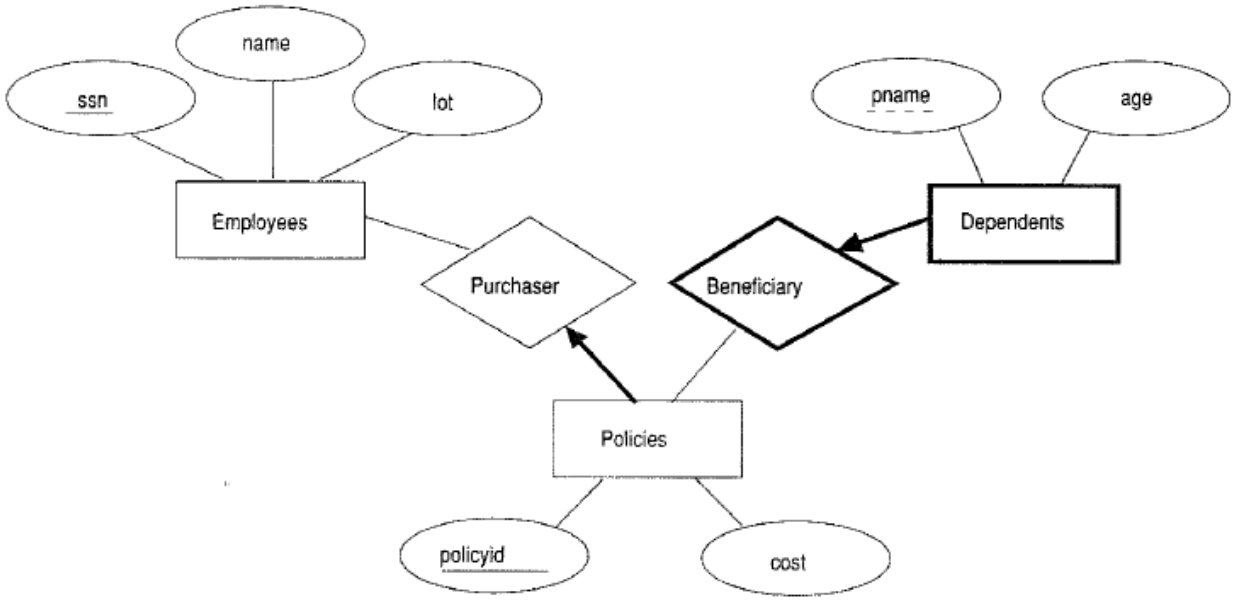
Ipotesi #1: relazione ternaria con vincoli

- Il secondo requisito potrebbe essere implementato imponendo un vincolo di partecipazione totale su POLICY
  - Problema: questa soluzione funziona solo se ogni polizza copre almeno un familiare a carico



Ipotesi #2: due relazioni binarie

- In situazioni come questa, la soluzione migliore è quella di introdurre 2 relazioni binarie:
  - Questa soluzione evidenzia che ci sono due relazioni distinte a cui partecipa POLICIES
  - In questo modo di usa una relazione binaria come identificante l'entità debole, cosa che in alcune versione di ER è richiesto



Relazioni binarie o ternarie: sintesi

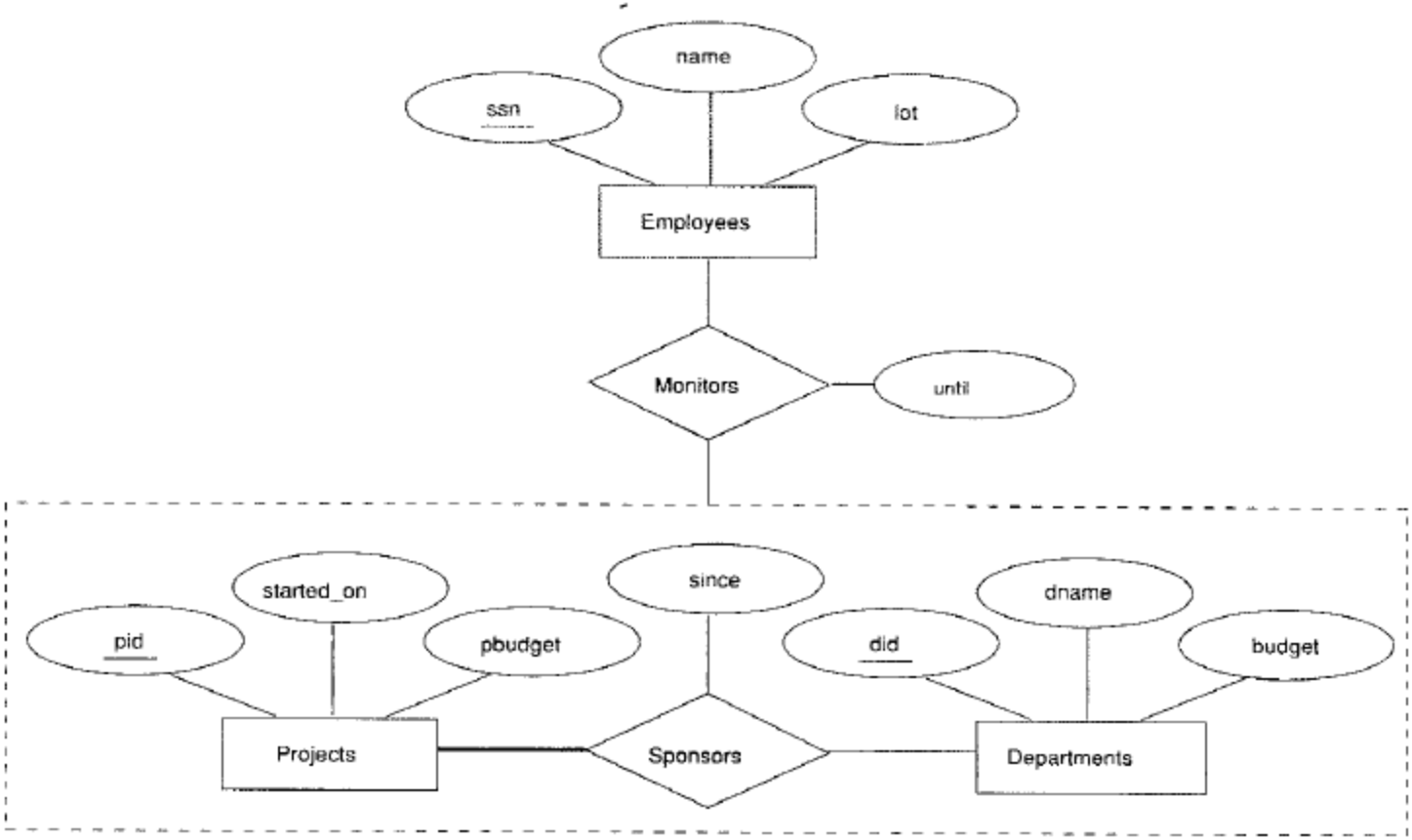
- La risposta non è univoca e va cercata in una comprensione profonda del significato dei requisiti
- Nell'esempio discusso la scelta preferibile è quella delle due relazioni binarie
- Nell'esempio della durata degli incarichi di gestione dei manager vista poco sopra, la scelta corretta è quella della relazione ternaria
- Spesso le decisioni di progettazione vanno riviste dopo una prima fase di elaborazione, con un processo a spirale che porta al modello finale

Aggregazione o relazione ternaria?

- Altra decisione piuttosto comune è se usare un'aggregazione o una relazione di grado 3
- La scelta è spesso guidata dall'informazione e dai vincoli che dobbiamo poter rappresentare nel modello
- Come esempio, riprendiamo il caso dell'impiegato che monitora un progetto sponsorizzato da un dipartimento e vediamo come alcune variazioni possono portarci a preferire una soluzione piuttosto che l'altra

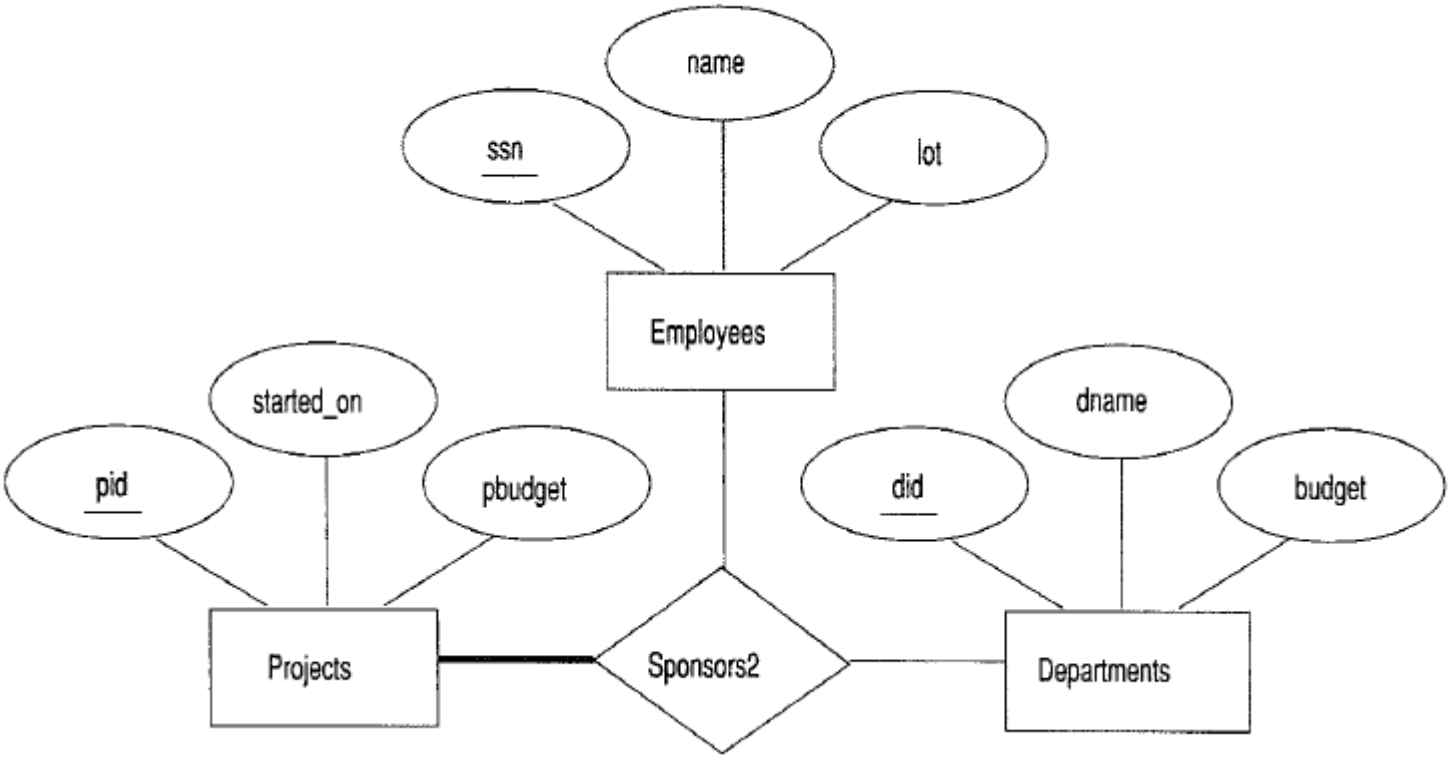
Aggregazione o relazione ternaria - Esempio

- Questa la soluzione presentata in precedenza (aggregazione):



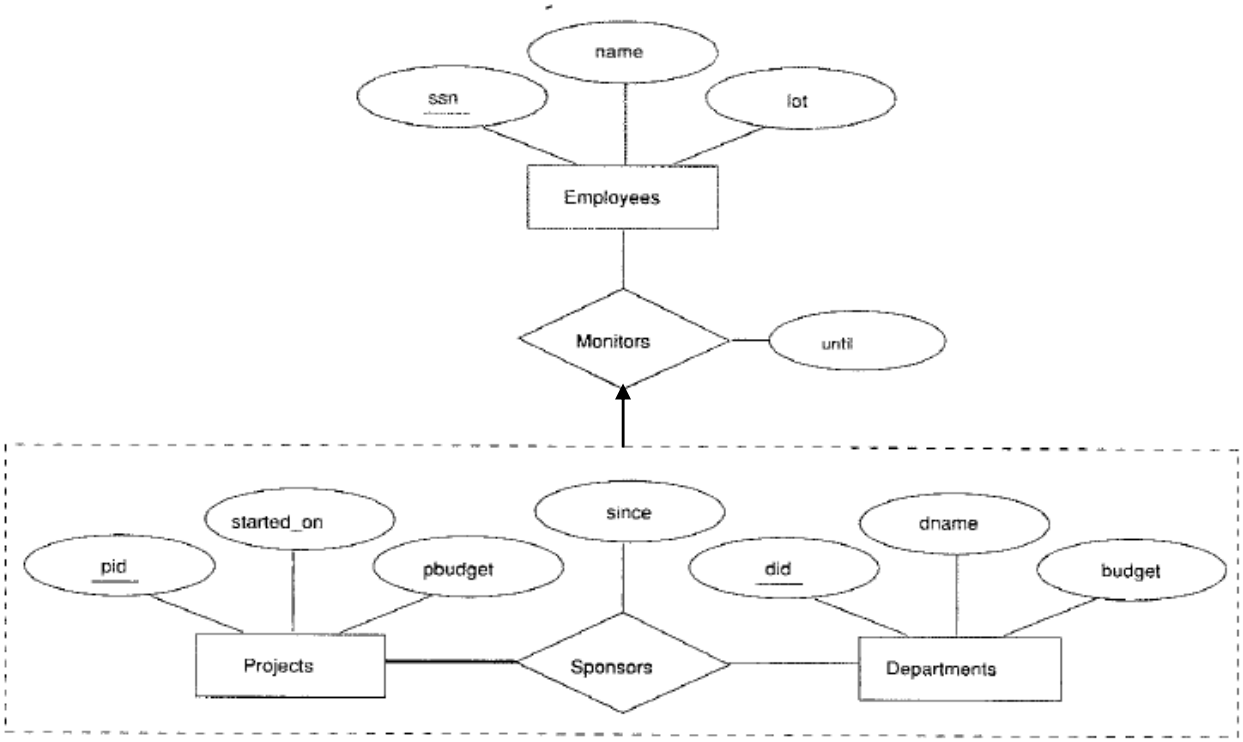
Aggregazione o relazione ternaria - Esempio

- Se non ci fosse bisogno di specificare il termine del monitoraggio (UNTIL), anche una relazione ternaria potrebbe andare bene:



Aggregazione o relazione ternaria - Esempio

- Tuttavia, la relazione ternaria SPONSORS2 non ci permette di catturare un requisito in cui si chiede che ogni sponsorizzazione sia monitorata al massimo da un impiegato, mentre MONITORS ce lo consente facilmente



DAI REQUISITI AL MODELLO: LINEE GUIDA

Come procedere dai requisiti al modello

1. Analisi del linguaggio naturale

- **Identificazione delle entità**: Cerca nomi di persone, luoghi, oggetti, concetti o eventi che ricorrono nei requisiti. Questi elementi spesso rappresentano le entità.
- **Individuazione dei verbi**: I verbi spesso indicano azioni o stati che collegano le entità, suggerendo potenziali relazioni tra di esse.
- **Riconoscere gli attributi**: Gli aggettivi o le frasi che descrivono le caratteristiche delle entità possono essere considerati attributi.

Come procedere dai requisiti al modello

2. Chiedere chiarimenti al cliente

- Se il significato di alcune parti del requisito non è chiaro, chiedi chiarimenti.
- Non inventare niente. Se per esempio il cliente non ha esplicitato un attributo o un vincolo, discutine con il cliente stesso e solo dopo eventualmente aggiungilo
- Questo assicura che le entità e le relazioni identificate riflettano accuratamente le esigenze del cliente.

Come procedere dai requisiti al modello

3. Identificazione delle entità

- **Nomi e sostantivi concreti**: Le entità sono spesso rappresentate da nomi propri, sostantivi concreti o concetti chiave (es. Cliente, Prodotto, Ordine)
- **Generalizzazione**: Se trovi più sostantivi simili o correlati, cerca di generalizzare per trovare una categoria comune (es. se ci sono "Macchina" e "Moto", potresti identificarle come "Veicolo")

Come procedere dai requisiti al modello

4. Individuazione delle relazioni

- **Relazioni esplicite**: Cerca frasi che indicano un'associazione tra due o più entità (es. «Un cliente effettua un ordine» o «Un dipendente afferisce a un dipartimento»)
- **Cardinalità**: considera la frequenza con cui un'entità può essere collegata ad un'altra. Per esempio, frasi come «ogni cliente può effettuare più ordini» indicano una relazione uno-a-molti
- **Partecipazione**: frasi del tipo «ogni dipartimento deve avere un manager» spesso indicano un vincolo di partecipazione (in questo caso totale)

Come procedere dai requisiti al modello

5. Identificazione degli attributi

- **Descrittori delle entità**: Ogni entità ha delle proprietà o attributi che la descrivono. Questo sono spesso indicate da frasi del tipo «Ogni cliente ha un nome, un indirizzo e un contatto».
- **Chiavi primarie**: Identifica attributi che sono unici per l'entità e che quindi possano fungere da chiavi primarie. Talvolta, le chiavi sono indicate da frasi del tipo «Ogni cliente è identificato con un ID»), altre volte va capito quale può essere una chiave dalla semantica dell'attributo.

Come procedere dai requisiti al modello

6. Validazione delle entità e relazioni

- **Completezza**: Assicurati che tutte le entità e le relazioni necessarie siano state identificate e che il modello sia completo rispetto ai requisiti
- **Ridondanza**: Evita duplicazioni inutili. Ogni entità e relazione dovrebbe essere unica a meno che non ci siano buone ragioni per includere duplicati. Un semplice esempio di ridondanza (sbagliata) è la duplicazione del nome del cliente nell'anagrafica e negli ordini.

Come procedere dai requisiti al modello

7. Iterazione e raffinamento

- **Feedback**: Presenta il modello in forma di diagramma ER al cliente e raccogli feedback per eventuali modifiche o aggiustamenti.
- **Raffinamento**: Modifica il modello in base ai feedback, iterando fino a ottenere una rappresentazione accurata dei requisiti.