Modellazione Dimensionale e Star Schema

Progettare Data Warehouse per Analisi Veloci e Intuitive

Unità IA.4 Lezione 2 Data Warehouse

Fact Tables: Misure del Business

Cosa Contengono

- **Misure numeriche** del business
- Metriche **quantitative** osservabili
- **Foreign keys** verso le dimensioni
- Dati a livello di **dettaglio** (grain)

Caratteristiche Chiave

- **Centro** dello Star Schema
- **Molto grandi** (milioni/miliardi record)
- **Grain** ben definito e rispettato
- Aggiornate **frequentemente**
- Contengono **fatti osservabili**

Esempio: Fact Vendite

```
fact_vendite
├── vendita_id (PK)
├──
├── -- Foreign Keys --
├── tempo_key (FK → dim_tempo)
├── prodotto_key (FK → dim_prodotto)
├── cliente_key (FK → dim_cliente)
├── canale_key (FK → dim_canale)
├──
├── -- Misure Numeriche --
├── quantita (INTEGER)
├── ricavo (DECIMAL)
├── costo (DECIMAL)
├── sconto (DECIMAL)
└── profitto (DECIMAL)
```

Dimension Tables: Contesto Descrittivo

Cosa Contengono

- **Attributi descrittivi** del business
- Contesto **testuale** e categorico
- **Primary key** (surrogate key)
- **Natural key** per riconciliazione

Caratteristiche Chiave

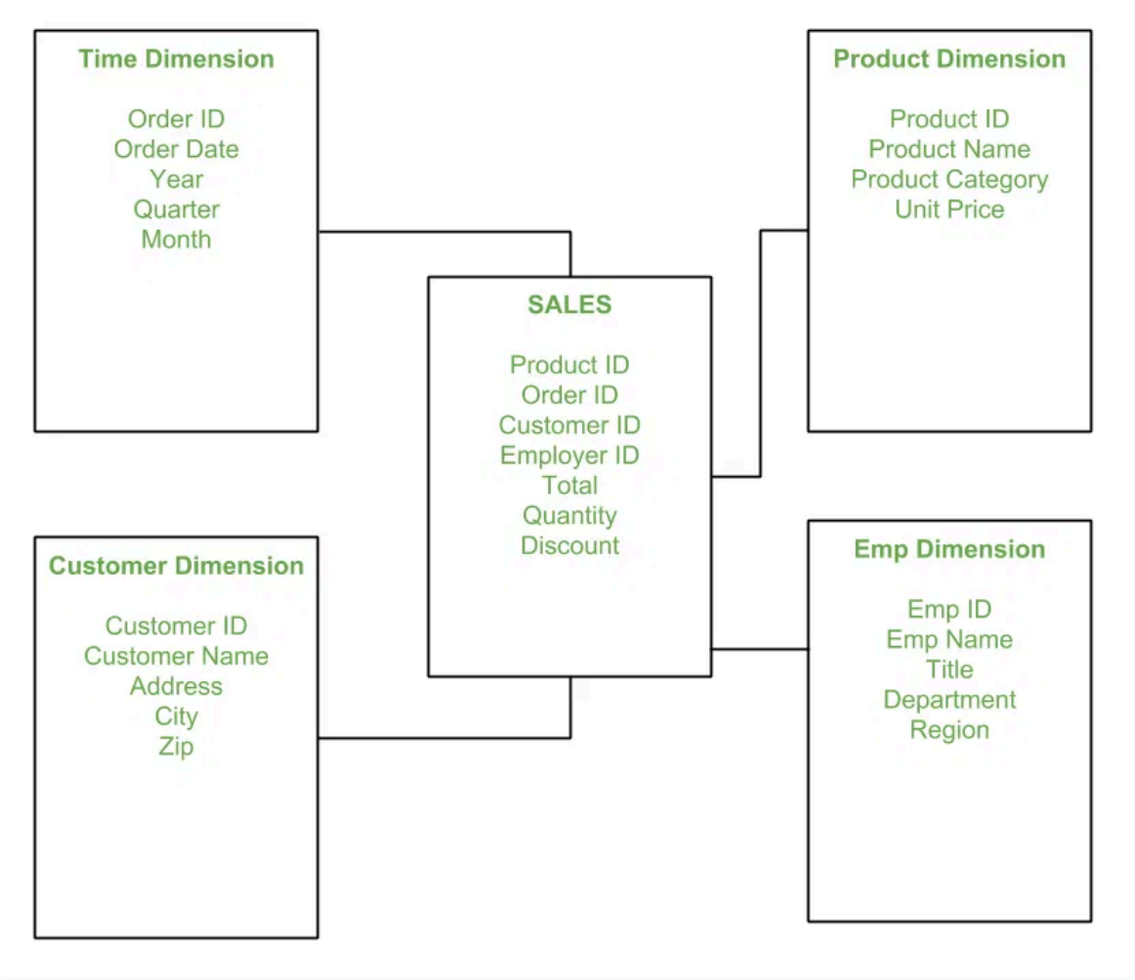
- **Raggi** dello Star Schema
- **Relativamente piccole** (migliaia record)
- **Denormalizzate** (gerarchie appiattite)
- Aggiornate **raramente**
- Forniscono il **contesto** per l'analisi

Esempio: Dimension Prodotto

```
dim_prodotto
├── prodotto_key (PK, surrogate)
├── prodotto_id (business key)
├──
├── -- Attributi Descrittivi --
├── nome_prodotto (TEXT)
├── descrizione (TEXT)
├──
├── -- Gerarchia Denormalizzata --
├── categoria (TEXT)
├── sottocategoria (TEXT)
├──
├── -- Altri Attributi --
├── marca (TEXT)
├── fornitore (TEXT)
├── prezzo_listino (DECIMAL)
└── data_inserimento (DATE)
```

Star Schema: Struttura e Vantaggi

Il Modello Dimensionale Più Semplice ed Efficace



Lo **Star Schema** è il modello dimensionale più diffuso: una **Fact Table centrale** circondata da **Dimension Tables** collegate direttamente, formando una struttura a stella.

Vantaggi Principali

Performance Elevate

Pochi JOIN necessari (2-4 tipicamente)

Semplicità Intuitiva

Schema facile da capire per utenti business

Flessibilità Analitica

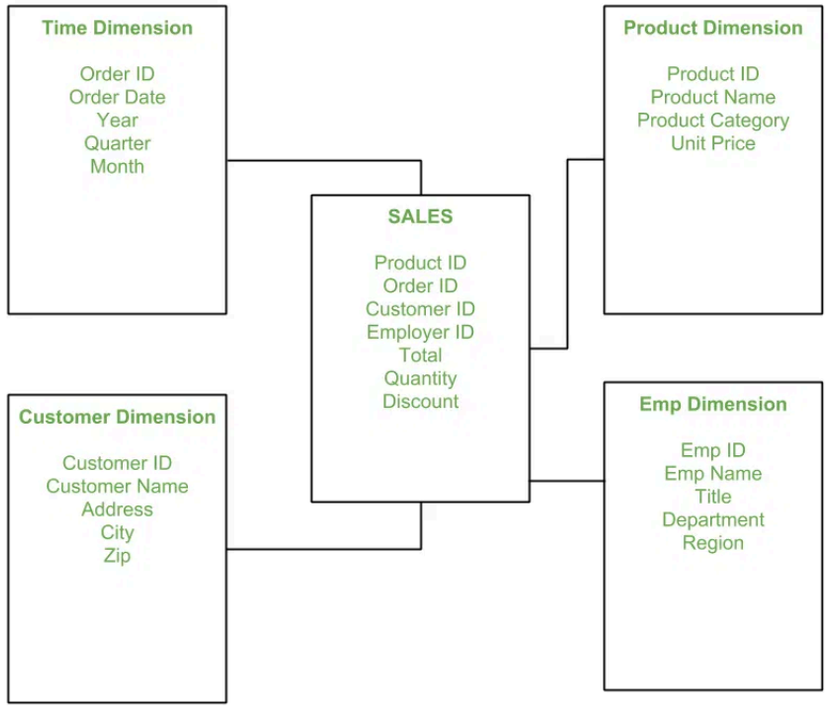
Supporta qualsiasi combinazione di dimensioni

Facile Estensione

Aggiungere dimensioni o misure è semplice

Star Schema TechStore: Struttura

E-commerce con Fact Vendite e 4 Dimensioni



Fact Table Centrale

fact_vendite con misure numeriche (quantità, ricavo, costo, profitto)

4 Dimensioni

dim_tempo, dim_prodotto, dim_cliente, dim_canale

Star Schema TechStore: Query Semplificata

Confronto OLTP vs Star Schema

</> Query Esempio: Ricavi Mensili per Categoria

```
SELECT
  t.anno,
  t.mese,
  p.categoria,
  SUM(f.ricavo) AS ricavi_totali
FROM
fact_vendite
f
JOIN
dim_tempo
t
  ON f.tempo_key = t.tempo_key
JOIN
dim_prodotto
p
  ON f.prodotto_key = p.prodotto_key
WHERE t.anno = 2024
GROUP BY t.anno, t.mese, p.categoria
ORDER BY t.mese, ricavi_totali DESC;

-- Solo 2 JOIN! Categoria già in dim_prodotto (denormalizzata)
```

OLTP Normalizzato

6

JOIN necessari

Star Schema

2

JOIN necessari

Il Grain: La Decisione Più Importante

DEFINIZIONE

Il **grain** (granularità) è la definizione di business di cosa rappresenta **una singola riga** nella fact table.

> *Il grain è la descrizione dell'evento di misurazione nel mondo fisico che dà origine a una misurazione.*
>
> — Ralph Kimball, Regola d'Oro

Esempi Corretti

- ✓ **Vendite retail:** Il "beep" dello scanner alla cassa
- ✓ **E-commerce:** Una riga per ogni prodotto in un ordine
- ✓ **Call center:** Una chiamata di assistenza
- ✓ **Web analytics:** Un click su una pagina

Esempi Sbagliati

- ✗ **"Vendite giornaliere"** — Troppo aggregato, perde dettaglio
- ✗ **"Ordini e prodotti"** — Ambiguo, non chiaro
- ✗ **"Transazioni > €100"** — È un filtro, non un grain
- ✗ **"Clienti e prodotti"** — Non descrive un evento

Le Prime Due Regole per il Grain

1

Scegli il Grain PIÙ BASSO Possibile

Il grain più dettagliato offre massima flessibilità di analisi e supporta il maggior numero di dimensioni. Puoi sempre aggregare, ma non puoi disaggregare.

✓ **Meglio:** *"Prodotto venduto" → "Ordine" → "Vendite giornaliere"*

2

Dichiaralo PRIMA di Tutto

Il grain deve essere definito prima di decidere quali dimensioni includere, quali misure calcolare, o come strutturare le tabelle. È la decisione di design più importante.

Ordine corretto: *Grain → Dimensioni → Misure → Struttura*

Regole per Scegliere il Grain (continua)

3

Sii SPECIFICO e Preciso

Descrivi l'evento di misurazione nel mondo fisico, non una lista tecnica di foreign keys. Usa linguaggio di business chiaro e comprensibile.

✗ **Sbagliato:** *"Vendite"* ✓ **Corretto:** *"Una riga per ogni prodotto venduto in un ordine"*

⚠ Resta FEDELE al Grain

Tutte le misure nella fact table devono essere **coerenti** con il grain dichiarato. Una misura a livello "ordine" non può stare in una fact table con grain "prodotto venduto"!

Tipi di Misure: Classificazione per Additività

Additive

DEFINIZIONE

Sommabili attraverso **tutte** le dimensioni

CARATTERISTICHE

Le più comuni e utili. Supportano aggregazioni complete e flessibili.

ESEMPI

- Quantità venduta
- Ricavo
- Costo
- Profitto
- Numero transazioni

Operazioni

SUM, AVG, MIN, MAX
su tutte le dimensioni

Semi-Additive

DEFINIZIONE

Sommabili su **alcune** dimensioni, non tutte

CARATTERISTICHE

Tipicamente saldi o livelli. NON sommabili attraverso il tempo.

ESEMPI

- Saldo conto bancario
- Livello inventario
- Numero dipendenti attivi
- Stock disponibile

Operazioni

AVG nel tempo
SUM su altre dimensioni

Non-Additive

DEFINIZIONE

NON sommabili attraverso **nessuna** dimensione

CARATTERISTICHE

Tipicamente rapporti o percentuali. Devono essere ricalcolate.

ESEMPI

- Prezzo unitario
- Percentuale sconto
- Margine %
- Tasso conversione
- Ratio qualsiasi

Best Practice

Memorizza componenti
Calcola a query time

I Tre Tipi di Fact Tables

Secondo la Metodologia Kimball

Transaction

GRAIN

Un evento che accade in un singolo istante

CARATTERISTICHE

- Evento **atomico**
- Tempo **imprevedibile**
- Sparsità irregolare
- Mai aggiornata
- Molto grande

QUANDO USARLA

- Massimo dettaglio
- Eventi irregolari
- Analisi singoli eventi

Esempio:
Vendita prodotto, Click web, Transazione bancaria

Periodic Snapshot

GRAIN

Misurazioni prese a intervalli regolari predefiniti

CARATTERISTICHE

- Intervallo **fisso**
- Densità **prevedibile**
- Sempre presente
- Mai aggiornata
- Media grandezza

QUANDO USARLA

- Snapshot regolari
- Garantire completezza
- Analisi trend

Esempio:
Saldi giornalieri, Inventario mensile, Metriche web giornaliere

Accumulating Snapshot

GRAIN

Un processo con inizio e fine ben definiti

CARATTERISTICHE

- Workflow **completo**
- **Multiple date**
- Record aggiornati
- Overwriting strategy
- Relativamente piccola

QUANDO USARLA

- Processo con fasi
- Analisi tempi lag
- Stato corrente

Esempio:
Elaborazione ordine, Gestione reclamo, Pipeline vendita

Dimension Tables e Surrogate Keys

Surrogate Keys vs Natural Keys

Aspetto	Natural Key	Surrogate Key
Origine	Sistema sorgente	Data Warehouse
Esempio	"LAPTOP-001"	12345
Significato	Ha significato business	Nessun significato
Stabilità	Può cambiare	Immutabile
Dimensione	Variabile (stringa)	Piccola (intero)

Perché Usare Surrogate Keys?

- ✓ **Stabilità:** Se il business key cambia, il surrogate key rimane
- ✓ **Performance:** Intero piccolo (4 byte) vs stringa lunga
- ✓ **SCD:** Gestire storico versioni (Slowly Changing Dimensions)
- ✓ **Integrazione:** Unificare dati da sistemi diversi

Denormalizzazione: Gerarchie Appiattite

```
-- Gerarchia: Categoria → Sottocategoria → Prodotto
-- TUTTO in una sola tabella!

dim_prodotto
├── prodotto_key (PK, surrogate)
├── prodotto_id (natural key)
├── nome_prodotto
├──
├── categoria ← Livello 1
├── categoria_desc
├── sottocategoria ← Livello 2
├── sottocategoria_desc
├──
├── marca
├── fornitore
└── prezzo_listino

-- Categoria ripetuta per ogni prodotto!
-- Ridondanza OK per performance
```

Best Practice: Memorizza ENTRAMBI i tipi di chiavi nella dimension table! Usa il surrogate key per i JOIN, ma mantieni il natural key per riconciliazione e debugging.

Star Schema vs Snowflake Schema

Aspetto	⭐ Star Schema	❄️ Snowflake Schema
Struttura Dimensioni	✅ Denormalizzate (flat)	❌ Normalizzate (gerarchiche)
Numero Tabelle	✅ Poche (1 fact + N dim)	❌ Molte (dimensioni divise)
Complessità Query	✅ Semplici (pochi JOIN)	❌ Complesse (molti JOIN)
Performance Query	✅ Veloci	❌ Più lente
Spazio Disco	⚠️ Maggiore (ridondanza)	✅ Minore (no ridondanza)
Manutenzione	✅ Più semplice	❌ Più complessa
Comprensibilità	✅ Intuitivo per utenti	❌ Richiede esperienza SQL
ETL	⚠️ Più complesso	✅ Più semplice

💡 Raccomandazione

Usa **Star Schema come default**. Lo spazio disco è economico, ma le query lente e complesse costano tempo e produttività. Snowflake Schema è utile solo in casi molto specifici (dimensioni enormi, vincoli di storage estremi).

Metodologia Kimball: 4 Passi

Come Progettare uno Star Schema Efficace

1 Scegli il Business Process

Identifica l'area di business da analizzare. Deve essere un processo misurabile e ben definito.

TechStore: *"Vendite e-commerce" (processo: cliente acquista prodotti online)*

2 Dichiara il Grain

Definisci cosa rappresenta una singola riga nella fact table. Questa è la decisione più importante!

TechStore: *"Una riga per ogni prodotto venduto in un ordine"*

3 Identifica le Dimensioni

Elenca tutti i modi in cui vuoi analizzare i dati. Ogni dimensione risponde a "Chi? Cosa? Dove? Quando?"

TechStore: *dim_tempo (quando?), dim_prodotto (cosa?), dim_cliente (chi?), dim_canale (dove?)*

4 Identifica le Misure

Determina i valori numerici da misurare. Devono essere coerenti con il grain dichiarato.

TechStore: *quantita, ricavo, costo, profitto, sconto_applicato*

Importante: L'ordine dei passi è fondamentale! Non puoi identificare le dimensioni prima di dichiarare il grain, e non puoi scegliere le misure prima di conoscere le dimensioni.

Errori Comuni da Evitare

1. Grain Non Chiaro o Ambiguo

Non dichiarare esplicitamente il grain o usare descrizioni vaghe come "vendite" invece di "prodotto venduto in un ordine". Questo porta a confusione, misure incompatibili, e decisioni di design sbagliate.

> ✓ **Soluzione:** Scrivi il grain in una frase chiara e specifica. Documentalo. Fallo approvare dagli stakeholder prima di procedere con dimensioni e misure.

2. Misure che Violano il Grain

Includere misure a livello "ordine" (es. spese_spedizione, numero_prodotti_totali) in una fact table con grain "prodotto venduto". Questo crea ridondanza e risultati incorretti nelle aggregazioni.

> ✓ **Soluzione:** Ogni misura deve essere coerente con il grain dichiarato. Se hai bisogno di misure a livelli diversi, crea una seconda fact table con grain appropriato.

3. Normalizzare le Dimensioni Inutilmente

Creare tabelle separate per categoria, sottocategoria, marca invece di denormalizzare tutto in dim_prodotto. Questo porta a Snowflake Schema con query complesse e performance peggiori.

> ✓ **Soluzione:** Appiattisci le gerarchie nelle dimension tables. La ridondanza è OK! Privilegia semplicità e performance delle query. Lo spazio disco è economico.

Errori Comuni da Evitare (continua)

4. Usare Solo Natural Keys

Usare codici prodotto o ID clienti del sistema sorgente come chiavi primarie invece di surrogate keys. Questo crea problemi quando i codici cambiano, quando si integrano sistemi diversi, o quando si gestisce lo storico (SCD).

> ✓ **Soluzione:** Genera sempre surrogate keys (interi sequenziali) come chiavi primarie delle dimension tables. Mantieni i natural keys come attributi per riconciliazione e debugging, ma non usarli per i JOIN.

5. Mescolare Processi Business Diversi

Combinare vendite online e resi, oppure ordini e spedizioni, in una singola fact table con grain confuso. Questo porta a misure incompatibili, NULL values eccessivi, e query complesse.

> ✓ **Soluzione:** Un business process = una fact table. Crea fact_vendite e fact_resi separate con grain specifici. Se necessario, uniscile nelle analisi con UNION o viste. Mantieni la purezza del modello.

Riepilogo e Prossimi Passi

Checklist Progettazione Star Schema

- ✓ Scegliere il business process da modellare
- ✓ Dichiarare il grain in modo specifico e preciso
- ✓ Identificare tutte le dimensioni di analisi
- ✓ Identificare le misure coerenti con il grain
- ✓ Creare fact table con foreign keys alle dimensioni
- ✓ Denormalizzare le dimension tables (gerarchie appiattite)
- ✓ Usare surrogate keys per tutte le dimensioni
- ✓ Mantenere anche i natural keys per riconciliazione

Principi Fondamentali

Il Grain è Sacro

Dichiaralo prima di tutto e resta fedele. Tutte le decisioni successive dipendono da questa scelta.

Denormalizzazione è OK

Ridondanza nelle dimensioni migliora performance e semplicità. Lo spazio disco è economico.

Semplicità Vince

Star Schema batte Snowflake. Query semplici e veloci sono più importanti di pochi MB risparmiati.

→ Prossimi Passi