



Relational Database

- conceived by IBM's E. F. Codd in early 1970, based on mathematics foundation called set theory and predicate logic.
- IBM began work on **SYSTEM/R**, which later became **SQL/DS** and then **DB2**.
- Oracle® shipped the first commercial relational database system, Oracle, in 1978.
- **Design goal** - overcome the problem of database **rigidity**, and be able to **respond quickly** to changing needs.



Benefit of Relational Design

Some benefits of relational database design :

- Data **entry, updates** and **deletions** will be efficient.
- Data **retrieval, summarization** and **reporting** will also be efficient.
- Behaves **predictably**.
- Information is stored in database rather than in application, the database is **self-documenting**.
- Changes to the database schema are easy to make.



Terminology

Relation

- A relation is a table with columns and rows.
- Perception of logical structure of the database, not the physical structure.
- Relation represents “things” (or **entities**) in the real world. It can be real-world objects or events, e.g, customer, order detail, product.

customer(custid, custname, companyno, address,
contact, region)

order_detail(orderno, productno, amount, price,
discount_given)



Terminology

Attribute

- A named column of a relation. Stores information pertaining to the entity.
- For example, branch relation having attributes for
bno (branch number)
street
city
pcode (postcode)
telno (telephone number)
faxno (fax number)



Terminology

- **Domain** Set of allowable values for one or more attributes. (eg. Range & type of value)
- **Tuple** A row of a relation.
- **Degree** Number of attributes in a relation.
- **Cardinality** Number of tuples in a relation.
- **Relation Database** A collection of normalized relations.



Alternative Terminology

	Data Processing Terms	
Formal Terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

- Formal terms are mostly used for technical users.
- Data processing terms are more natural for non technical users.

CUSTOMER RECORD	
Name:	Bart Simpson
Address:	100 Main St. Seattle, WA 98122
Phone:	322-3424

Diagram illustrating the mapping of a customer record to a database table structure.

FIELDS

Arrows indicate the mapping from the fields in the **CUSTOMERS TABLE** to the corresponding fields in the **CUSTOMER RECORD**:

- Name** maps to the Name field.
- Address** maps to the Address field.
- Phone** maps to the Phone field.

Name	Address	Phone
Bart Simpson	100 Main St. Seattle, WA 98122	322-3424
John Handley	2 2nd Street Bellevue, WA 98055	789-3421
George Chan	11023 115th Ct. Kirkland, WA 98033	827-3344



Properties of Relations

A relation has the following characteristics:

- **Relation name** has a name that is **distinct** from all others relations.
- Each cell of relation contains exactly one **atomic** (single) value.
- Each **attribute** has a **distinct** name.
- Values of an attribute are all from the **same domain**.
- **Order** of attributes has **no significance**.
- Each tuple is distinct; there are **no duplicate tuples**.
- **Order** of tuples has **no significance**, theoretically.



Relational Keys

Superkey An attribute or a set of attributes that uniquely identifies a tuple within a relation.

Candidate Key A superkey (K) such that no proper subset is a superkey within the relation. In each tuple of R , the values of K uniquely identifies that tuple.

No proper subset of K has the uniqueness property.

Primary Key Candidate key selected to identify tuples **uniquely** within relation.

Alternate Key Candidate keys that are not selected to be the primary key.



Relational Keys

Foreign Key An attribute or set of attributes within one relation that matches candidate key of some (possibly same) relation.



Relational Integrity

Null Represents a value for an attribute that is currently unknown or is not applicable for this tuple.

- Nulls are a way to deal with incomplete or exceptional data.
- Not same as zero or spaces; zeros and spaces are values but a null represents the absence of a value.

Entity Integrity In a base relation, no attribute of a primary key can be null.



Relational Integrity

Referential Integrity

— If foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.

Enterprise Constraints

Additional rules specified by users or database administrator.

- *Triggering Operations*

other rules to protect the validity of attribute values; should be removed from application programs and incorporated into a **repository** in the dbms.



Steps in Designing a Database

Here are the steps in the database design process :

Step 1: Determine the purpose of your database.

- This will help you decide which facts you want to store within the database(abstraction).

Step 2: Determine the tables you need.

- This step involves dividing your information into separate subjects, such as "Employees" or "Orders." Each subject will be a table in your database.



Steps in Designing a Database

Step 3: Determine the fields you need.

- Decide the information you want to keep in each table. For example, one field in an Employees table could be Last Name; another could be Hire Date.
- **Step 4: Determine the relationships.**
- Look at each table and decide how the data in one table is related to the data in other tables.
- Add fields to tables or create new tables to clarify the relationships, as necessary.



Steps in Designing a Database

Step 5: Refine your design.

- Analyze your design for errors.
- Create the tables and add a few records of sample data.
- Test to get the results you want from your tables.
- Make adjustments to the design as needed.



Step 1: Determine the Purpose

- The first step in designing a database is to determine the **purpose** & **usage** of the database.
- This tells you what **information** you want from the database.

What do you need to do at this stage:

- **Talk** to the people who will use the database.
- **Brainstorm** about the questions you'd like the database to answer.
- Sketch out the **reports** you'd like it to produce.
- Gather the **forms** you currently use to record your data.



Example - Northwind Traders

Tracking Sales and Inventory

Northwind Traders, an import/export company that sells speciality foods from around the world, wants a database that can track information about the company's sales and inventory.

Possible Questions Asked:

- How many sales of our featured product did we make last month?
- Where do our best customers live?
- Who's the supplier for our best-selling product?

Products On Order

14-Jun-98

Category Name	Product Name	In Stock	On Order	Supplier Name	Phone
Beverages	18% off total Inventory				
	Chai	39	0	Exotic Liquids	(71) 555-2222
	Chang	17	40	Exotic Liquids	(71) 555-2222
	ChartreuseVerte	69	0	Aux joyeux ecclésiastiques	(1) 03.83.00.68
	Côte de Blaye	17	0	Aux joyeux ecclésiastiques	(1) 03.83.00.68
	Guaraná Fantástica	20	0	Refrescos Americanos SDA	(11) 555 4640
	Ipi-chi Coffee	17	10	Leka Trading	555-8787
	Lakkolakkol	57	0	Karkki Oy	(93) 10956
	Laughing Lumberjack Lager	52	0	Bigfoot Breweries	(93) 555-9931
	Outback Lager	15	10	Pavlova, Ltd.	(03) 444-2343
	Rhinobrau Klosterbier	125	0	Pluspar Lebensmittelgroßhdlg	(069) 992755
	Sasquatch Ale	111	0	Bigfoot Breweries	(93) 555-9931
	Steeleye Stout	20	0	Bigfoot Breweries	(93) 555-9931
		539	60		
Condiments	18% of Total Inventory				
	Aniseed Syrup	13	70	Exotic Liquids	(71) 555-2222

Northwind Traders

Example of Report & Form

ORDER FORM

Northwind Traders

One Northwind Way, Twin Points, VA, 98099
Phone: 1-206-555-8888 Fax: 1-206-555-8889

Bill To: Save-a-lot Markets
187 Suffolk Ln.
Boise ID 83720

Ship To: Save-a-lot Markets
187 Suffolk Ln.
Boise ID 83720

Date of Order: 14-Jun-98

Product ID	Product Name	Unit Price	Quantity	Extended Price
16	Pavlova	\$17.45	55	\$959.75
24	Guaraná Fantástica	\$4.50	20	\$90.00
26	Inland Sil	\$19.00	40	\$760.00
Subtotal:				\$1,809.75
Freight				\$211.22
Total:				\$2,020.97



Step 2: Determine the Tables

- To categorize the information into tables.
- Analyze the input forms & reports to determine the entity that we are interested in.
- Northwind order form as an example. It includes facts about the customer -- the *customer's address* and *phone number* -- along with facts about the *order*.
- Need careful analysis so that will not run into problems.

Problem: If stored the customer facts in the same table as the order facts:

Which address is correct?

Orders : Table				
	Order ID	Order Date	Ship Name	Ship Address
▶	11063	27-Apr-95	Hungry Owl All-Night Grocers	8 Johnstown Road
	10985	27-Mar-95	Hungry Owl All-Night Grocers	28 Johnstown Road
	10736	08-Nov-94	Hungry Owl All-Night Grocers	8 Johnstown Road
	10321	30-Sep-93	Island Trading	Garden House
	10315	23-Sep-93	Island Trading	Garden House
Record: 14 345 of 830				

For accuracy, store each fact only once.

Customers : Table				
	Customer ID	Company Name	Address	City
▶	HUNGO	Hungry Owl All-Night Grocers	8 Johnstown Road	Cork
	ISLAT	Island Trading	Garden House	Cowes
	KOENE	Königlich Essen	Maubelstr. 90	Brandenburg
	LACOR	La corne d'abondance	67, avenue de l'Europe	Versailles
	LAMAI	La maison d'Asie	1 rue Alsace-Lorraine	Toulouse
	LAUGB	Laughing Bacchus Wine Cellars	1900 Oak St.	Vancouver
Record: 14 37 of 91				



Table Design Problems

1. Introducing Errors in Duplicate Information.

- Suppose that one customer places three different orders.
- We could add the customer's address and phone number to your database three times.
- Multiplies the chance of data entry errors.

2. Deleting Valuable Information.

- A new customer places an order and then cancels.
- If we delete the order from the table, we would delete the customer's name and address as well.
- We want to keep this new customer record for sending catalog.



Strategy For Table Design

- Look at the information we want and divide it into fundamental subjects you want to track, such as customers, employees, products, services provided...
- Each of these subjects is a **candidate** for a separate table.
- The Northwind Traders order form and Products On Order report include information about these subjects:

Customers

Products

Suppliers

Orders



Step 3: Determining the Fields

- Decide what we want to know about the people, things, or events recorded in the table.
- Fields is the characteristics of the table.
- Each record (or row) in the table contains the same set of fields or characteristics.



Tips for Determining Fields

1. **Relate each field directly to subject of the table.**
 - A field that describes the subject of a different table belongs in the other table.
 - Later, define relationships between tables to combine the data from fields in multiple tables.
2. **Do not include derived or calculated data.**
 - In most cases, do not store the result of calculations in tables.



Tips for Determining Fields

3. Include all the information we need.

- It's easy to overlook important information.
- Return to the information we gathered in the first step of the design process.
- Relook into the paper forms and reports to make sure all the information we required in the past is included in tables or can be derived from them.



Tips for Determining Fields

4. Store information in its smallest logical parts.

- **Avoid single field** for full names.
e.g. product names along with product descriptions
full address...
- If we combine more than one kind of information in a field, will be difficult to retrieve individual facts later.
- Try to break down information into logical parts.
- Example, create separate fields for first and last name, break product name into category and description, break address into street, postcode, city, state...



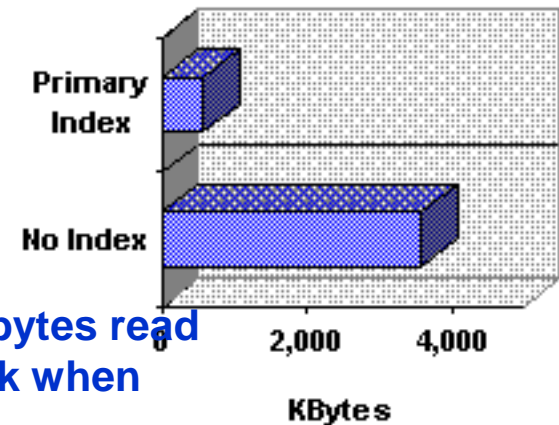
Primary Key Fields

- In order for database to work most efficiently, each table in your database should include a field or set of fields that uniquely identifies each individual record stored in the table.
- This is often a unique identification number, such as an employee ID number or a serial number.

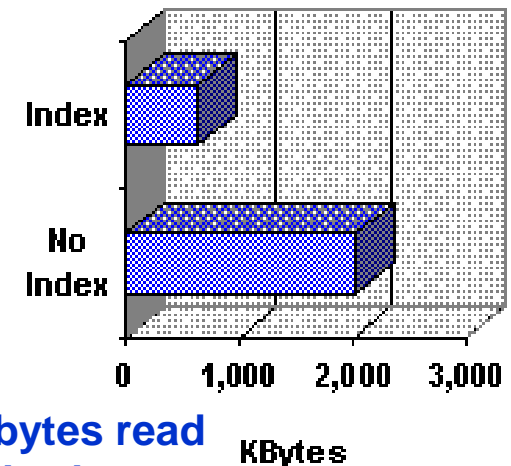
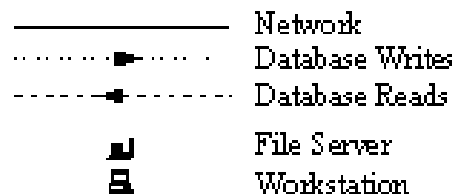
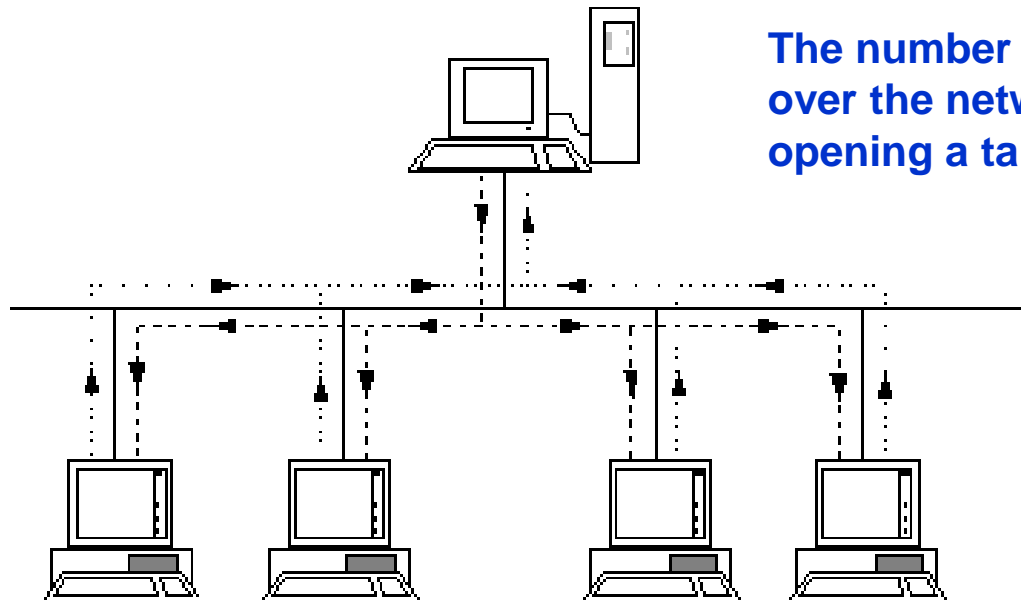
Performance Comparison with / without Primary Key

Example Database Information

Number of tables in database	1
Number of records in the table	19,991
Maximum record size for the table	283 bytes
Average record size for the table	120 bytes
Database size when compacted	3,473,408 bytes



The number of bytes read over the network when opening a table.



The number of bytes read over the network when executing a quick sort.



Criteria for Primary Key

- There's no absolute rule as to which candidate key is best.
- Fabian Pascal, in his book *SQL and Relational Basics*, notes that the decision should be based upon 3 principles of
 - **Minimality**, choose the fewest columns necessary.
 - **Stability**, choose a key that seldom changes.
 - **Simplicity/Familiarity**, choose a key that is both simple and familiar to users.



Choosing Primary Key

- **Does not allow duplicate or null values** in a primary key field. We should not choose a primary key that could contain such values.
- Primary key field is used to look up records, it **should not be too long to remember or type**. You may want it to have a certain number of letters or digits, or be in a certain range.
- The size of the primary key affects the **speed** of operations in database, we can set a property to limit the size of the field.
- For best performance, **use the smallest size** that will accommodate the values we need to store in the field.



Primary Key - Northwind Traders

The primary key of the Northwind Products table contains product ID numbers.

Primary key

	Product ID	Product Name	Units In Stock
▶	54	Tourtière	21
	55	Pâté chinois	115
	56	Gnocchi di nonna Alice	21
	57	Ravioli Angelo	36
	58	Escargots de Bourgogne	62

Record: 14 54 of 77

No two product numbers are the same ...

... but other fields may contain duplicate values.



Step 4: Determining the Relationships

- Now we have divided information into tables, we need a way to bring tables back together in meaningful ways.
- Setting up relationship amongst tables.

Information in this form comes from the Customers table...

... the Employees table...

... the Orders table ...

Orders

Bill To: Richter Supermarkt
Grenzacherweg 237
Genève 1203
Switzerland

Ship To: Richter Supermarkt
Starenweg 5
Genève 1204
Switzerland

Salesperson: Davolio, Nancy

Ship Via: ☒ Speedy ☐ United ☐ Federal

Order ID: 10537 **Order Date:** 11-May-96 **Required Date:** 25-May-96 **Shipped Date:** 16-May-96

	Product:	Unit Price:	Quantity:	Discount:	Extended Price:
▶	Röd Kaviar	\$15.00	9	0%	\$135.00
	Mozzarella di Giovanni	\$34.80	21	0%	\$730.80
	Escargots de Bourgogne	\$13.25	20	0%	\$265.00
	Manjimup Dried Apples	\$53.00	6	0%	\$318.00

Subtotal: \$1,823.80
Freight: \$78.85
Total: \$1,902.65

Print Invoice

Record: 14 620 of 830

... the Products table ...

... and the Order Details table.



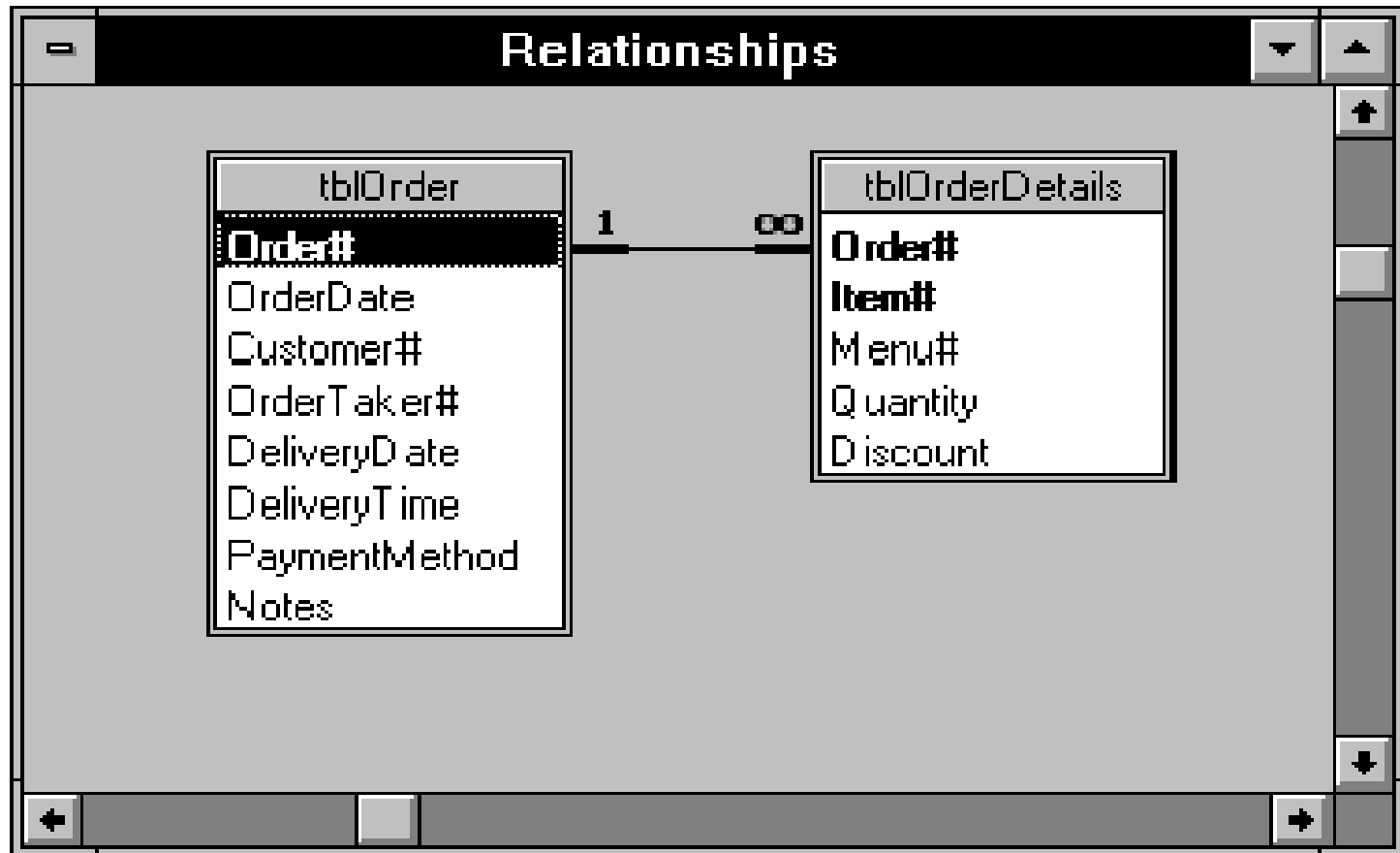
Types of Relationship

- To set up the relationship correctly, you must first determine the nature of the relationship.
- There are three types of relationships between tables:
 1. One-to-Many relationships
 2. Many-to-Many relationships
 3. One-to-One relationships



One-to-Many Relationship

- Most common type of relationship in a relational database.
- Two tables are related in a one-to-many (1–M) relationship if for every row in the first table, there can be zero, one, or many rows in the second table,
- But for every row in the second table there is exactly one row in the first table.
- Also referred to as a **Parent-child** or **Master-detail** relationship.



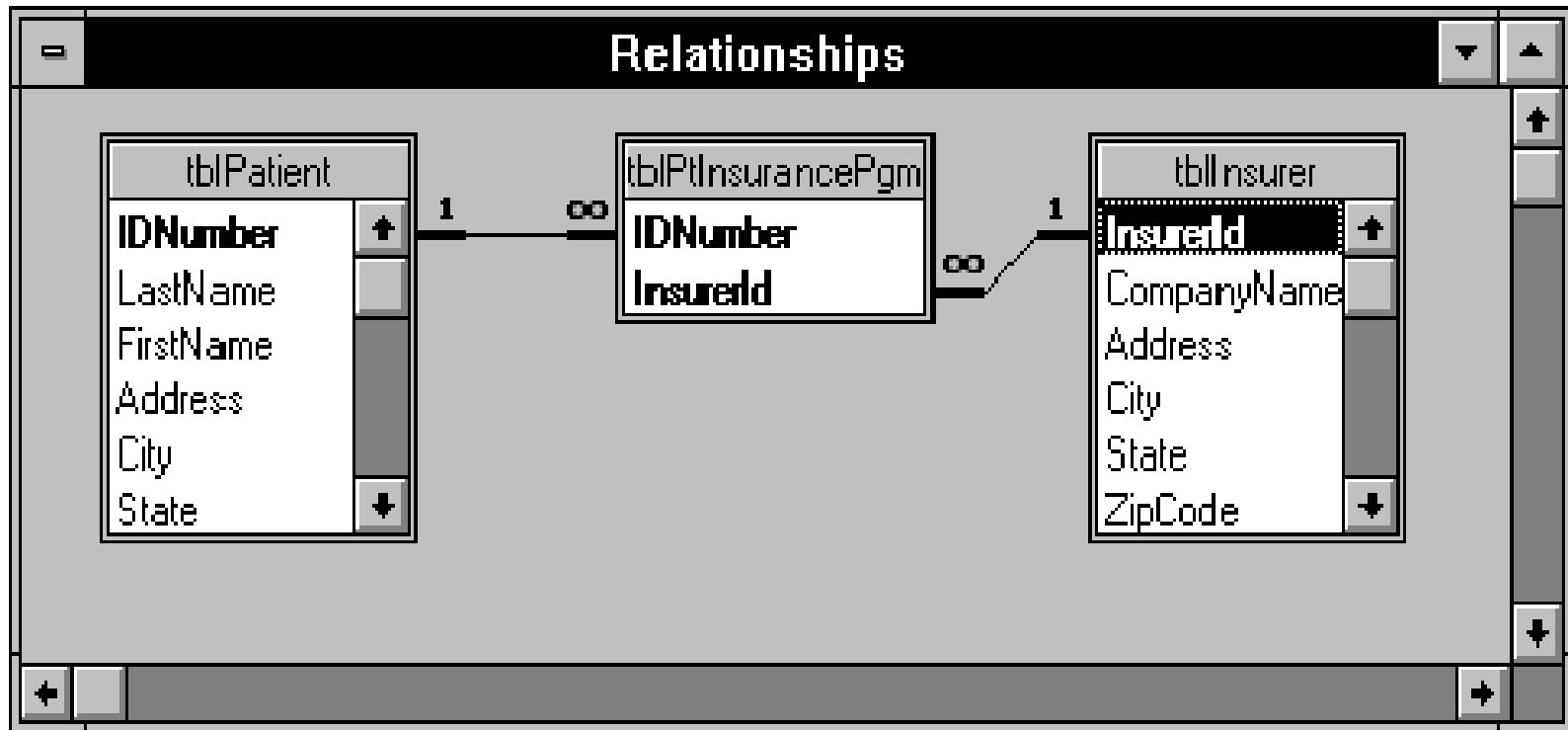
One-Many Relationship

For example, each order for a pizza delivery business can have multiple items. Therefore, **tblOrder** is related to **tblOrderDetails** in a one-to-many relationship.



Many-to-Many Relationship

- Two tables are related in a many-to-many (M–M) relationship when for every row in the first table, there can be many rows in the second table.
- And for every row in the second table, there can be many rows in the first table.
- Can not be directly modeled in most RDBMS.
- The relationships must be broken into multiple one-to-many relationships, by creating a third linking table.
- Often put the primary key from each of the two tables into the linking table.



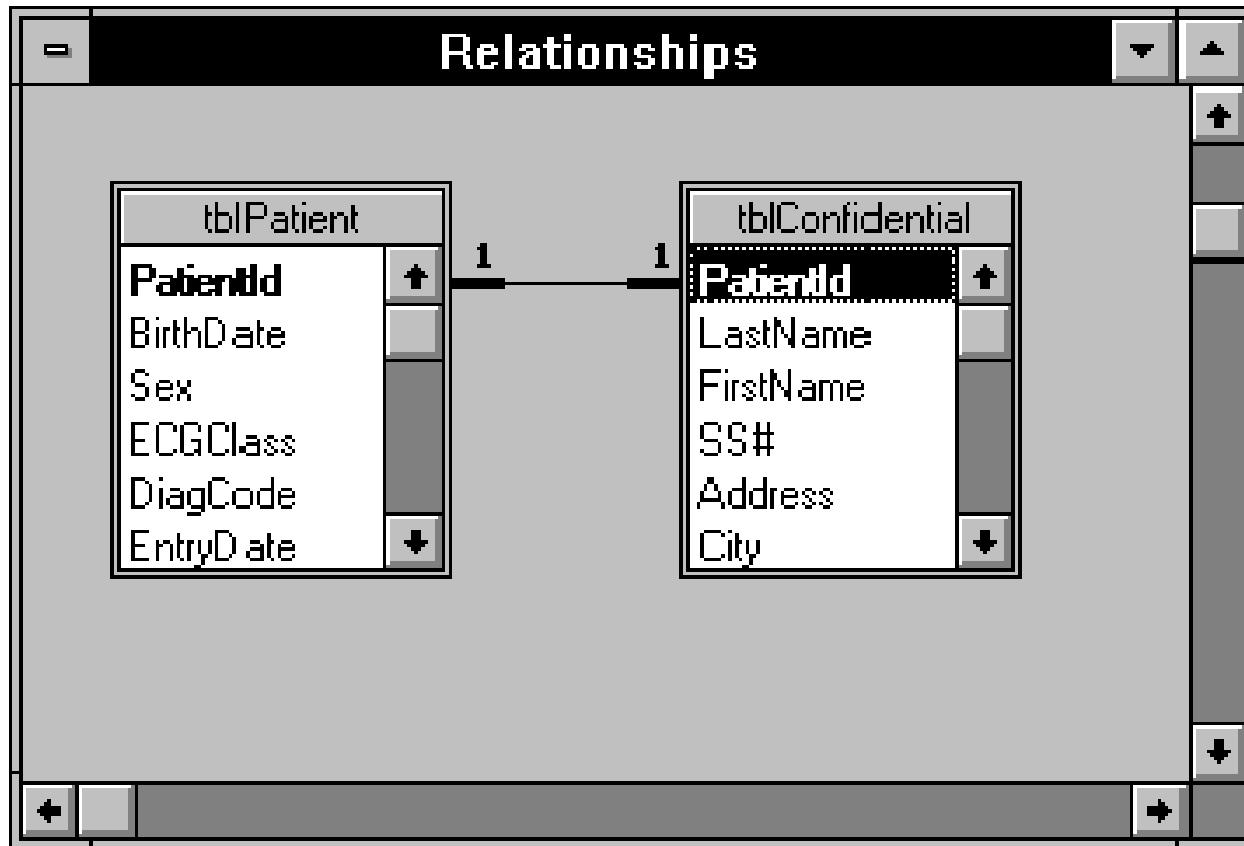
Many-many Relationship

- Example, a patient may be covered by multiple insurance plans and a given insurance company covers multiple patients.
- The **tblPatient** table in a medical database would be related to the **tblInsurer** table in a many-to-many relationship.
- In order to model the relationship between these two tables, we create a third, linking table, called **tblPtInsurancePgm** that contain a row for each insurance program under which a patient was covered



One-to-One Relationship

- Two tables are related in a one-to-one (1–1) relationship if, for every row in the first table, there is at most one row in the second table.
- Seldom occur in the real world.
- Often created to get around some limitation of the database management software.
- In Microsoft Access, we might have to split a table into two or more tables due to **security** or **performance concerns** or because of the **limit** of 255 columns per table.
- Tables have the same primary key, which will serve as the join column.



One-One Relationship

For example, we might keep most patient information in **tblPatient**, but put especially sensitive information (e.g., patient name, social security number and address) in **tblConfidential**.



Step 5: Refining the Design

Study the design and detect any flaws that might remain:

Ask yourself these question:

- Did you forget any fields?
- Did you choose a good primary key for each table?
- Are you repeatedly entering duplicate information in one of your tables?
- Do you have tables with many fields, a limited number of records, and many empty fields in individual records?
- If so, think about redesigning the table so it has fewer fields and more records.



CARTESIAN PRODUCT

Is a mathematical concept to understand RELATION

Assume that the sets D1 and D2 where

$$D1 = \{2,4\}$$

$$D2 = \{1,3,5\}$$

Cartesian product of these two sets can be written as

$$D1 * D2 = \{(2,1),(2,3),(2,5),((4,1),(4,3),(4,5))\}$$

A subset of these Cartesian product is a relation

For example,

$$R = \{(2,1),(4,1)\}$$



CARTESIAN PRODUCT

$$R = \{(2,1), (4,1)\}$$

The relation R includes all those pairs in which the second element is ONE.

$$R = \{(x,y) \mid x \in D1, y \in D2, \text{ and } y = 1\}$$

We could form another relation S, in which the first element is always twice the second.

$$S = \{(x,y) \mid x \in D1, y \in D2, \text{ and } x = 2y\}$$

Example :

$$S = \{(2,1)\}$$



CARTESIAN PRODUCT

Example :

$$D1 = \{1,3\} \quad D2 = \{2,4\} \quad D3 = \{(5,6)\}$$

$$D1 * D2 * D3 = \{(1,2,5), \dots\dots\dots\}$$

This can be written as $\prod_{i=1}^n D_i$



DATABASE RELATION

RELATION SCHEMA :

A relation name followed by a set of attribute and domain name pairs

Example :

Let A_1, A_2, \dots, A_n be attributes with domain $D_1, D_2, D_3, \dots, D_n$

Thus, the Relation R is a set of n-tuples $(A_1 : d_1, A_2 : d_2, \dots, A_n : d_n)$ such that $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$



DATABASE RELATION

RELATION SCHEMA :

A relation name followed by a set of attribute and domain name pairs

Example :

Let A_1, A_2, \dots, A_n be attributes with domain $D_1, D_2, D_3, \dots, D_n$

Thus, the Relation R is a set of n-tuples $(A_1 : d_1, A_2 : d_2, \dots, A_n : d_n)$ such that $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$



RELATIONAL ALGEBRA

There are a variety of languages used by the RDBMS for manipulating relations

Types

PROCEDURAL

NON PROCEDURAL

Relational Algebra and Relational Calculus as defined by Codd (1971) as the basis of relational languages



RELATIONAL ALGEBRA

Informally we can say that Relational Algebra as a procedural language : used to tell the DBMS, how to build a relation from one or more relations in the database.

Is used to measure the selective power of relational language.



RELATIONAL ALGEBRA

Relational Algebra is a theoretical language with operations that work on one or more relations to define another relation without changing the original relations

Possible Operations in Relational Algebra :

Selection

Projection

Cartesian product

Union and

Set Difference



RELATIONAL ALGEBRA

In addition to that there are also

**JOIN
INTERSECTION &
DIVISION**

**The SELECTION and PROJECTION operations are
UNARY operations, since they operate on only one
relation**

The others are BINARY operations



SELECTION

σ predicate(R)

The selection operator works on a single relation R and defines a relation that contains only those tuples [rows] of R that satisfy the specified condition (predicate)

Example : $\sigma_{\text{salary} > 1000} (\text{STAFF})$



PROJECTION

$\Pi_{\text{col1, col2, ..., colN(R)}}$

The projection operator works on a single relation R and defines a relation that contains a vertical subset of R extracting the values of specified attributes and eliminating duplicate.

Example : $\Pi_{\text{sno, fname, lname, salary (STAFF)}}$



CARTESIAN PRODUCT

The Cartesian Product operator defines a relation that is the concatenation of every tuple of relation R with every tuples of relation S.

Example

$\prod_{\text{rno, fname, lname}} (\text{RENTAR}) \times$

$\prod_{\text{rno, pno, comment}} (\text{VIEWING})$



What is it ?

$\sigma_{\text{renter.no} = \text{viewing.rno}} \left(\Pi_{\text{rno}, \text{fname}, \text{lname}}(\text{RENTAR}) \times \Pi_{\text{rno}, \text{pno}, \text{comment}}(\text{VIEWING}) \right)$



UNION

R U S

The UNION of relation R and S with I and J tuples, respectively, is obtained by concatenating them into one relation with a maximum of (I + j) tuples, duplicate tuples being eliminated. R and S must be Union-compatible.

$$\prod_{\text{area(BRANCH)}} \cup \prod_{\text{area (PROPERTY_RENT)}}$$



SET DIFFERENCE R - S

The **DIFFERENCE** operator defines a relation consisting of the tuples are in relation R, but not in S. R and S must be **UNION** compatible.

$$\Pi_{\text{city}}(\text{BRANCH}) - \Pi_{\text{city}}(\text{PROPERTY_RENT})$$



JOIN OPERATOR

Is one of the operation in Relational Algebra used to combine two or more relation to form a new relation

Types

THETA - JOIN

EQUI-JOIN

NATURAL JOIN

OUTER JOIN

SEMI-JOIN



THETA - JOIN R \Leftrightarrow_F S

The Theta Join operator defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.

$$R \Leftrightarrow_F S = \sigma_f(R \times S)$$

RENTER \Leftrightarrow renter.rno = viewing.rno **VIEWING**



NATURAL- JOIN $R \Leftrightarrow S$

The Natural Join is an equi join of the two relation R and S over all common attributes x. One occurrences of each common attribute is eliminated from the result.

$$R \Leftrightarrow S$$



SEMI- JOIN $R \Delta_F S$

The SEMI Join operator defines a relation
defines a relation that contains the tuples of R
that participate in the join or R with S

$$R \Delta_F S = \prod_A (R \Delta_F S)$$

A is the set of all attributes for R



INTERSECTION $R \cap S$

The INTERSECTION consist of the set of all tuples that are in both R and S. R and S must be union-compatible.

$$R \cap S = R - (R - S)$$



Division $R \div S$

The DIVISION consist of the set of all tuples from R defined over the attribute C that match the combination of **every** tuples in S.



CARTESIAN PRODUCT



Views

Base Relation

A named relation, corresponding to an entity in conceptual schema, whose tuples are physically stored in database.

View

Dynamic result of one or more relational operations operating on the base relations to produce another relation.



Views

- A view is a virtual relation that **does not actually exist** in the database but is produced upon request, at time of request.
- Contents of a view are defined as a query on one or more base relations. Any operations on the view are **automatically translated** into operations on underlying relations.
- Views are **dynamic**, meaning that changes made to base relations that affect view attributes are immediately reflected in the view.
- When users make **permitted changes** to view, changes are made to underlying relations.