

Note introduttive

Il metodo di lavoro proposto ripercorre il ciclo di vita di un sistema informativo automatizzato attraverso l'analisi, il progetto, la realizzazione e il testing.

Il sistema informativo è stato realizzato in Access impiegando l'implementazione dell'SQL nel RDBMS, per fornire la possibilità a tutti gli allievi di collaudare il sistema proposto.

In allegato a questa prova è anche possibile effettuare il download del file di database *PrenotazioniAlbergo.mdb*, realizzato in Access 2000 (ma collaudato anche con Access XP) che contiene:

1. le tabelle di base e le relazioni del database;
2. i servizi richiesti per l'albergo;
3. una maschera di avvio, che definisce (richiamando macro) una interfaccia grafica (riportata nella figura seguente) per l'utente per l'aggiornamento dei dati nelle tabelle di base e l'esecuzione delle operazioni del sistema;



4. un insieme di dati inseriti nelle tabelle di base per il testing dei servizi del sistema informativo.

Indice dell'esercitazione

ANALISI	3
1. SPECIFICHE PER I DATI.....	3
2. SPECIFICHE PER LE OPERAZIONI.....	3
3. SPECIFICHE TECNOLOGICHE	3
4. SPECIFICHE PER IL TESTING	3
PROGETTO CONCETTUALE	3
PROGETTO DELLO SCHEMA STATICO (MODELLO LOGICO DEI DATI)	3
<i>Fatti elementari</i>	3
<i>Tipi di entità e attributi</i>	4
<i>Le relazioni</i>	4
<i>Documentazione dello schema statico</i>	4
PROGETTO DELLO SCHEMA DINAMICO (LE OPERAZIONI).....	6
<i>Documentazione dello schema dinamico</i>	6
REALIZZAZIONE	7
PASSAGGIO DALLO SCHEMA STATICO DEL PROGETTO CONCETTUALE AL MODELLO RELAZIONALE	7
CREAZIONE DEL DATABASE.....	8
DIMENSIONAMENTO DEL DATABASE	9
CREAZIONE DEL DATABASE NEL RDBMS ACCESS	10
REALIZZAZIONE DEI SERVIZI	11
<i>Servizio: lista camere prenotate</i>	11
<i>Servizio: elenco prenotazioni clienti</i>	12
<i>Servizio: elenco disdette clienti</i>	13
<i>Servizio: numero clienti presenti in un dato giorno</i>	14

Analisi

1. Specifiche per i dati

Il sistema deve raccogliere e memorizzare tutti i dati relativi:

- ❑ ai **clienti**, che prenotano le camere;
- ❑ alle **camere**, prenotate dai clienti;
- ❑ alle **prenotazioni**, effettuate dai clienti per le camere.

2. Specifiche per le operazioni

Il sistema deve offrire i seguenti servizi:

1. la lista delle camere prenotate in un determinato giorno;
2. l'elenco ordinato dei clienti che in un determinato giorno hanno prenotato camere nell'albergo;
3. l'elenco ordinato dei clienti che hanno dato disdetta in un certo giorno dell'anno;
4. il numero di clienti presenti in un dato giorno.

3. Specifiche tecnologiche

Il sistema informativo centralizzato deve essere realizzato con un server di database, al quale possono accedere diversi utenti mediante personal computer, inseriti nella rete locale dell'albergo. Si ipotizza che l'albergo disponga di 40 camere, mentre si stima che il numero medio di clienti dell'albergo sia 1000. Il sistema deve essere in grado di gestire i dati relativi all'albergo per almeno un anno.

4. Specifiche per il testing

Per il testing del sistema informativo centralizzato, ipotizzare una serie di dati iniziali ed eseguire i servizi creati, verificando se i risultati ottenuti sono coerenti con lo stato della base di dati.

Progetto concettuale

Progetto dello schema statico (modello logico dei dati)

Fatti elementari

Sulla base dell'analisi del testo iniziale del problema e delle specifiche, il sistema può essere schematizzato nei seguenti fatti elementari.

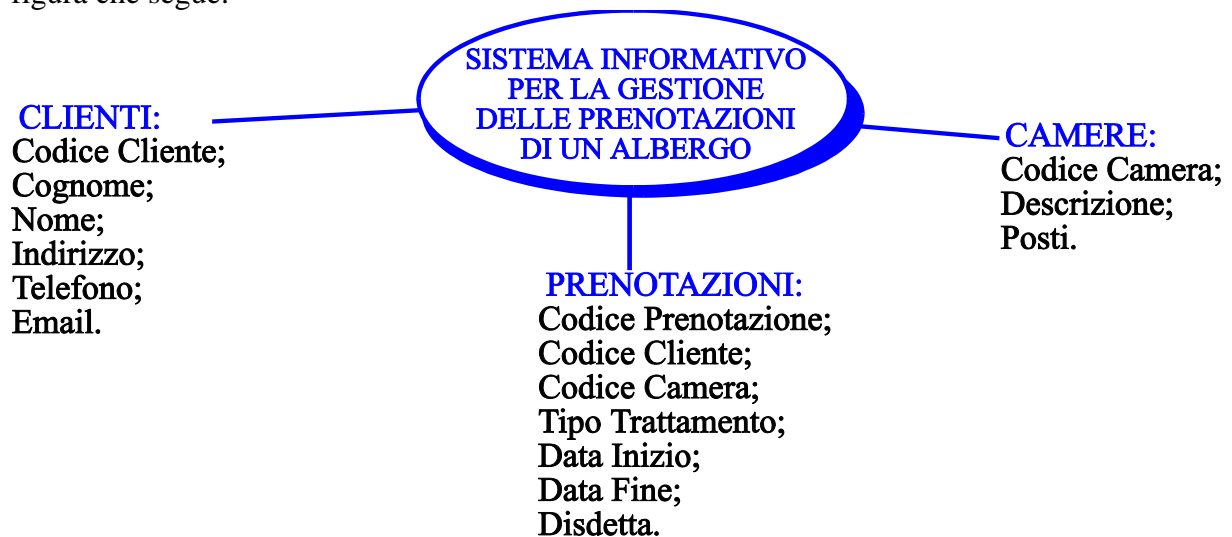
❑ Un cliente può effettuare più prenotazioni nell'albergo.
❑ Una prenotazione può riferirsi a più clienti che soggiornano nell'albergo.
❑ Ogni singola camera può avere più prenotazioni, in periodi diversi.

Tipi di entità e attributi

I tipi di entità caratteristici del sistema sono:

Clienti
Camere
Prenotazioni

I tipi di entità vengono individuati dai gruppi di attributi illustrati in modo dettagliato nella figura che segue.



Le relazioni

Le relazioni tra i tipi di entità nel modello dei dati sono individuate nel seguente diagramma Entità-Relazioni.



Nella fase di realizzazione del progetto concettuale la relazione molti a molti tra i tipi di entità *Clienti* e *Prenotazioni* dovrà essere trasformata in due associazioni uno a molti e molti a uno, introducendo un tipo di entità ausiliario.

Documentazione dello schema statico

PROGETTO CONCETTUALE: Gestione prenotazioni albergo;
INIZIO SCHEMA STATICO:

TIPO DI ENTITÀ: Clienti;

ATTRIBUTI:

Codice Cliente N \rightarrow N **TIPO:** Prenotazioni **ATTRIBUTO:** Codice Cliente;
Cognome;
Nome;
Indirizzo;
Telefono;

Email;

REGOLE:

Codice Cliente: chiave primaria;

Codice Cliente, Cognome, Nome e Telefono non possono assumere valori nulli;

L'attributo Codice Cliente ha il formato CC-XXXX (con XXXX cifre numeriche);

TIPO DI ENTITÀ: Camere;

ATTRIBUTI:

Codice Camera $\xrightarrow{1 \rightarrow N}$ TIPO: Prenotazioni **ATTRIBUTO:** Codice Camera;
Descrizione;
Posti;

REGOLE:

Codice Camera: chiave primaria;

Codice Camera, Descrizione e Posti non possono assumere valori nulli;

L'attributo Codice Camera è un numero intero;

TIPO DI ENTITÀ: Prenotazioni;

ATTRIBUTI:

Codice Prenotazione;

Codice Cliente $\xrightarrow{N \rightarrow N}$ TIPO: Clienti **ATTRIBUTO:** Codice Cliente;

Codice Camera $\xrightarrow{N \rightarrow 1}$ TIPO: Camere **ATTRIBUTO:** Codice Camera;

Tipo Trattamento;

Data Inizio;

Data Fine;

Disdetta;

REGOLE:

Codice Prenotazione: chiave primaria;

Disdetta è un valore booleano con valori limitati a Sì (True) o No (False);

Codice Prenotazione, Codice Cliente, Codice Camera, Tipo Trattamento, Data Inizio, Data Fine e Disdetta non possono assumere valori nulli;

Tipo Trattamento può assumere solo i valori "HB", "FB" e "B&B";

L'attributo Codice Prenotazione è un codice numerico intero che si incrementa automaticamente di una unità ad ogni inserimento di una nuova prenotazione;

Il tipo di entità Prenotazioni è soggetto al vincolo di integrità referenziale con il tipo Clienti, mediante l'attributo Codice Cliente (ogni prenotazione deve sempre fare riferimento ad un cliente già esistente);

Il tipo di entità Prenotazioni è soggetto al vincolo di integrità referenziale con il tipo Camere, mediante l'attributo Codice Camera (ogni prenotazione deve sempre fare riferimento ad una camera esistente dell'albergo);

FINE SCHEMA STATICO

Progetto dello schema dinamico (le operazioni)

Documentazione dello schema dinamico

In base agli obiettivi del problema, la documentazione delle operazioni è la seguente.

INIZIO SCHEMA DINAMICO:

OPERAZIONE: lista camere prenotate;

DESCRIZIONE: la lista delle camere a disposizione in un determinato giorno;

TIPO: Ricerca;

UTENTI: addetti alla reception, direttore albergo;

ARGOMENTI DI INGRESSO: data;

RISULTATI: Codice camera, Descrizione e Posti;

OPERAZIONE: elenco prenotazioni clienti;

DESCRIZIONE: l'elenco ordinato dei clienti che in un determinato giorno hanno prenotato camere nell'albergo;

TIPO: Ricerca, Ordinamento;

UTENTI: addetti alla reception, direttore albergo;

ARGOMENTI DI INGRESSO: data;

RISULTATI: Cognome, Nome, Telefono e Email dei clienti;

OPERAZIONE: elenco disdette clienti;

DESCRIZIONE: l'elenco ordinato dei clienti che hanno dato disdetta in un certo giorno dell'anno;

TIPO: Ricerca, Ordinamento;

UTENTI: addetti alla reception, direttore albergo;

ARGOMENTI DI INGRESSO: data;

RISULTATI: Cognome, Nome, Telefono, Email;

OPERAZIONE: numero clienti presenti in un dato giorno;

DESCRIZIONE: il numero totale dei clienti presenti in un determinato giorno dell'anno;

TIPO: Ricerca, Elaborazione;

UTENTI: addetti alla reception, direttore albergo;

ARGOMENTI DI INGRESSO: data;

RISULTATI: numero clienti presenti;

FINE SCHEMA DINAMICO

FINE PROGETTO.

Tra i servizi del sistema informativo, si dovranno anche realizzare tutte le operazioni di aggiornamento dei tipi di entità previsti nello schema statico.

Realizzazione

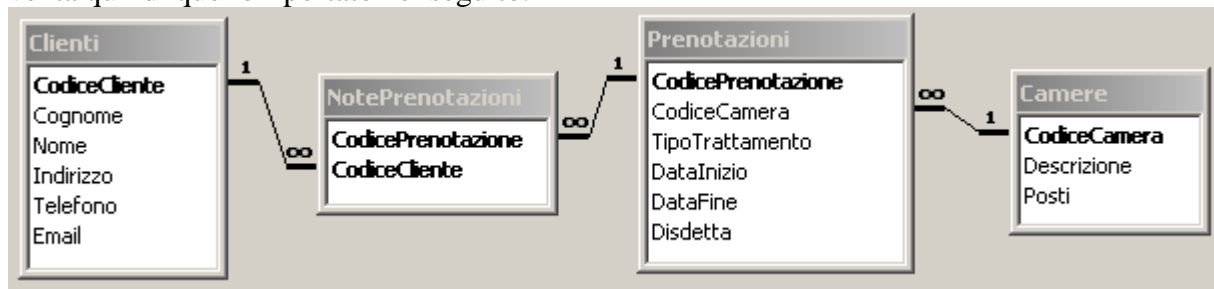
Passaggio dallo schema statico del progetto concettuale al modello relazionale

Per la creazione dello schema logico della base di dati relazionale effettueremo i seguenti due passi.

1. Traduzione dello schema statico concettuale in un modello relazionale preliminare.
2. Applicazione delle tre forme normali per verificare se lo schema preliminare può essere ulteriormente ottimizzato.

Nella fase di realizzazione del progetto concettuale, la relazione molti a molti tra *Clienti* e *Prenotazioni* deve essere trasformata in due associazioni uno a molti e molti a uno, introducendo una tabella ausiliaria *NotePrenotazioni*. Come suggerito nei fatti elementari del progetto concettuale, in ogni camera prenotata possono alloggiare uno oppure più clienti. La tabella *NotePrenotazioni* conterrà quindi il codice oppure i codici dei clienti che hanno prenotato (ed eventualmente occupano se non hanno dato disdetta) una camera. Con questa tabella ausiliaria, lo schema relazionale soddisfa le tre forme normali.

Lo schema logico relazionale del database per la gestione delle prenotazioni dell'albergo diventa quindi quello riportato nel seguito.



Creazione del database

Nel seguito è riportato il codice sorgente interpretato da un RDBMS (script), scritto nel linguaggio standard SQL, per la creazione dello schema logico del database.

```
CREATE DATABASE GestionePrenotazioniAlbergo
CREATE TABLE Clienti
(
    CodiceCliente CHAR(7) NOT NULL,
    Cognome       CHAR(15) NOT NULL,
    Nome          CHAR(15) NOT NULL,
    Indirizzo     CHAR(30) NOT NULL,
    Telefono      CHAR(12) NOT NULL,
    Email         CHAR(30),
    PRIMARY KEY (CodiceCliente),
    CHECK (CodiceCliente LIKE 'CC-____')
)
CREATE TABLE Camere
(
    CodiceCamera INTEGER NOT NULL,
    Descrizione  CHAR(50),
    Posti        INTEGER NOT NULL,
    PRIMARY KEY (CodiceCamera)
)
CREATE TABLE Prenotazioni
(
    CodicePrenotazione INTEGER IDENTITY(1,1),
    CodiceCamera       INTEGER NOT NULL,
    TipoTrattamento   CHAR(3) NOT NULL,
    DataInizio        DATE NOT NULL,
    DataFine           DATE NOT NULL,
    Disdetta           BIT DEFAULT 0,
    PRIMARY KEY (CodicePrenotazione),
    CHECK (TipoTrattamento IN ('HB', 'FB', 'B&B')),
    FOREIGN KEY (CodiceCamera) REFERENCES Camere (CodiceCamera)
)
CREATE TABLE NotePrenotazioni
(
    CodicePrenotazione INTEGER NOT NULL,
    CodiceCliente      CHAR(7) NOT NULL,
    PRIMARY KEY (CodicePrenotazione, CodiceCliente),
    FOREIGN KEY (CodicePrenotazione)
        REFERENCES Prenotazioni (CodicePrenotazione),
    FOREIGN KEY (CodiceCliente)
        REFERENCES Clienti (CodiceCliente)
)
```

Lo script deve essere eseguito in un RDBMS per la creazione delle tabelle e delle relazioni della base di dati.

Dimensionamento del database

La seguente tabella descrive il dimensionamento dei file fisici del database (file principale e registro delle transazioni), effettuato tenendo conto dei dati da archiviare.

Calcolo delle dimensioni occupate...	Elaborazione	Risultato
da ogni singola tabella del database utente	Numero righe (stima) tabella × Dimensione riga	Tabella Clienti = 1000 clienti × 109 byte = 109.000 byte
		Tabella Camere = 40 camere × 58 byte = 2.320 byte
		Tabella Prenotazioni = 365 giorni × 40 camere × 28 byte = 408.800 byte
		Tabella NotePrenotazioni = 4 clienti massimo × numero prenotazioni × 11 byte = 642.400 byte
dall'intero database utente	\sum Dimensione delle tabelle nel database	Tabella Clienti + Tabella Camere + Tabella Prenotazioni + Tabella NotePrenotazioni = 1.162.520 byte = 1.135 KB (1,1 MB)
dai database di sistema	1%÷5% × Dimensione iniziale del database utente	5% × 1.135 KB = 56,75 KB
Dimensione file principale di database =		Database utente + Database di sistema = 1.192 KB (1,16 MB)
dal registro delle transazioni	10%÷25% × Dimensione database utente	25% × 1.135 KB = 283,7 KB
Dimensione registro delle transazioni =		284 KB
dagli indici associati alla chiave primaria delle tabelle	Numero righe tabella × Dimensione chiave primaria	Tabella Clienti = 1000 × 7 byte = 7.000 byte Tabella Camere = 40 × 4 byte = 160 byte Tabella Prenotazioni = 14.600 × 4 byte = 58.400 byte Tabella NotePrenotazioni = 58.400 × 11 byte = 642.400 byte
Dimensione totale file indici =		707.960 byte = 691,4 KB (691 KB)
Totale file fisici del database =		1.192 KB + 284 KB + 691 KB = 2.167 KB (2,1 MB)

I calcoli precedenti sono stime nel caso peggiore (*worst case*) ottenute arrotondando i risultati in eccesso.

Creazione del database nel RDBMS Access

Lo schema logico può essere realizzato in Access (in alternativa);

- in modo interattivo, definendo le singole tabelle di base e impostando le relazioni;
- nella modalità programma, eseguendo nell'ordine i quattro script SQL, documentati nella tabella che segue. La sintassi standard SQL è stata adattata a quella implementata nel RDBMS Access.

Tabelle di base	Codice SQL nella modalità SQL Visualizzazione SQL																								
<table border="1"> <thead> <tr> <th colspan="3" style="background-color: #000080; color: white;">Clienti : Tabella</th> </tr> <tr> <th style="width: 5%;"></th> <th style="width: 70%;">Nome campo</th> <th style="width: 25%;">Tipo dati</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">🔑</td> <td>CodiceCliente</td> <td>Testo</td> </tr> <tr> <td></td> <td>Cognome</td> <td>Testo</td> </tr> <tr> <td></td> <td>Nome</td> <td>Testo</td> </tr> <tr> <td></td> <td>Indirizzo</td> <td>Testo</td> </tr> <tr> <td></td> <td>Telefono</td> <td>Testo</td> </tr> <tr> <td></td> <td>Email</td> <td>Testo</td> </tr> </tbody> </table>	Clienti : Tabella				Nome campo	Tipo dati	🔑	CodiceCliente	Testo		Cognome	Testo		Nome	Testo		Indirizzo	Testo		Telefono	Testo		Email	Testo	<p style="text-align: center;">Nome query: ScriptCreaTabellaClienti</p> <pre>CREATE TABLE Clienti (CodiceCliente CHAR(7) NOT NULL, Cognome CHAR(15) NOT NULL, Nome CHAR(15) NOT NULL, Indirizzo CHAR(30) NOT NULL, Telefono CHAR(12) NOT NULL, Email CHAR(30), PRIMARY KEY (CodiceCliente));</pre>
Clienti : Tabella																									
	Nome campo	Tipo dati																							
🔑	CodiceCliente	Testo																							
	Cognome	Testo																							
	Nome	Testo																							
	Indirizzo	Testo																							
	Telefono	Testo																							
	Email	Testo																							
<table border="1"> <thead> <tr> <th colspan="3" style="background-color: #000080; color: white;">Camere : Tabella</th> </tr> <tr> <th style="width: 5%;"></th> <th style="width: 70%;">Nome campo</th> <th style="width: 25%;">Tipo dati</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">🔑</td> <td>CodiceCamera</td> <td>Numerico</td> </tr> <tr> <td></td> <td>Descrizione</td> <td>Testo</td> </tr> <tr> <td></td> <td>Posti</td> <td>Numerico</td> </tr> </tbody> </table>	Camere : Tabella				Nome campo	Tipo dati	🔑	CodiceCamera	Numerico		Descrizione	Testo		Posti	Numerico	<p style="text-align: center;">Nome query: ScriptCreaTabellaCamere</p> <pre>CREATE TABLE Camere (CodiceCamera INTEGER NOT NULL, Descrizione CHAR(50), Posti INTEGER NOT NULL, PRIMARY KEY (CodiceCamera));</pre>									
Camere : Tabella																									
	Nome campo	Tipo dati																							
🔑	CodiceCamera	Numerico																							
	Descrizione	Testo																							
	Posti	Numerico																							
<table border="1"> <thead> <tr> <th colspan="3" style="background-color: #000080; color: white;">Prenotazioni : Tabella</th> </tr> <tr> <th style="width: 5%;"></th> <th style="width: 70%;">Nome campo</th> <th style="width: 25%;">Tipo dati</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">🔑</td> <td>CodicePrenotazione</td> <td>Contatore</td> </tr> <tr> <td></td> <td>CodiceCamera</td> <td>Numerico</td> </tr> <tr> <td></td> <td>TipoTrattamento</td> <td>Testo</td> </tr> <tr> <td></td> <td>DataInizio</td> <td>Data/ora</td> </tr> <tr> <td></td> <td>DataFine</td> <td>Data/ora</td> </tr> <tr> <td></td> <td>Disdetta</td> <td>Sì/No</td> </tr> </tbody> </table>	Prenotazioni : Tabella				Nome campo	Tipo dati	🔑	CodicePrenotazione	Contatore		CodiceCamera	Numerico		TipoTrattamento	Testo		DataInizio	Data/ora		DataFine	Data/ora		Disdetta	Sì/No	<p style="text-align: center;">Nome query: ScriptCreaTabellaPrenotazioni</p> <pre>CREATE TABLE Prenotazioni (CodicePrenotazione COUNTER, CodiceCamera INTEGER NOT NULL, TipoTrattamento CHAR(3) NOT NULL, DataInizio DATE NOT NULL, DataFine DATE NOT NULL, Disdetta LOGICAL, PRIMARY KEY (CodicePrenotazione), FOREIGN KEY (CodiceCamera) REFERENCES Camere (CodiceCamera));</pre>
Prenotazioni : Tabella																									
	Nome campo	Tipo dati																							
🔑	CodicePrenotazione	Contatore																							
	CodiceCamera	Numerico																							
	TipoTrattamento	Testo																							
	DataInizio	Data/ora																							
	DataFine	Data/ora																							
	Disdetta	Sì/No																							
<table border="1"> <thead> <tr> <th colspan="3" style="background-color: #000080; color: white;">NotePrenotazioni : Tabella</th> </tr> <tr> <th style="width: 5%;"></th> <th style="width: 70%;">Nome campo</th> <th style="width: 25%;">Tipo dati</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">🔑</td> <td>CodicePrenotazione</td> <td>Numerico</td> </tr> <tr> <td style="text-align: center;">🔑</td> <td>CodiceCliente</td> <td>Testo</td> </tr> </tbody> </table>	NotePrenotazioni : Tabella				Nome campo	Tipo dati	🔑	CodicePrenotazione	Numerico	🔑	CodiceCliente	Testo	<p style="text-align: center;">Nome query: ScriptCreaTabellaNotePrenotazioni</p> <pre>CREATE TABLE NotePrenotazioni (CodicePrenotazione INTEGER NOT NULL, CodiceCliente CHAR(7) NOT NULL, PRIMARY KEY (CodicePrenotazione, CodiceCliente), FOREIGN KEY (CodicePrenotazione) REFERENCES Prenotazioni (CodicePrenotazione), FOREIGN KEY (CodiceCliente) REFERENCES Clienti (CodiceCliente));</pre>												
NotePrenotazioni : Tabella																									
	Nome campo	Tipo dati																							
🔑	CodicePrenotazione	Numerico																							
🔑	CodiceCliente	Testo																							

Gli schemi delle tabelle di base del database sono stati poi completati in modo interattivo (nel file di database *PrenotazioniAlbergo.mdb*) definendo i vincoli di integrità dei dati e quelli referenziali sulle relazioni.

Realizzazione dei servizi

I servizi del sistema informativo sono stati realizzati in Access, per cui verrà presentata l'interrogazione scritta nell'implementazione SQL di questo RDBMS (che presenta alcune differenze di sintassi rispetto allo standard SQL) e la corrispondente query creata impiegando la tecnica QBE (*Query By Examples*). Le interrogazioni SQL sono state collaudate digitando direttamente il codice in un oggetto *Query* di Access nella modalità *Visualizzazione SQL*.

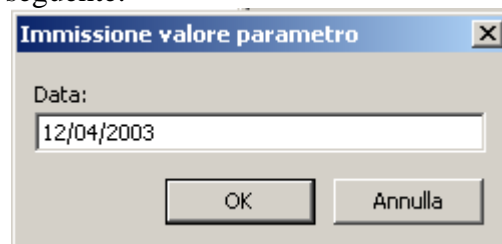
Servizio: lista camere prenotate

(Nome query: *ListaCamerePrenotate*)

Codice sorgente SQL

```
SELECT Camere.CodiceCamera, Camere.Descrizione, Camere.Posti
FROM Camere INNER JOIN Prenotazioni
ON Camere.CodiceCamera = Prenotazioni.CodiceCamera
WHERE (Prenotazioni.Disdetta = No) AND
(Prenotazioni.DataInizio <= [Data:]) AND
([Prenotazioni].[DataFine] >= [Data:]);
```

Nel codice precedente, così come in quelli dei servizi successivi, [Data:] è un parametro di ingresso il cui valore viene richiesto da Access prima dell'esecuzione dell'operazione con una finestra di dialogo del tipo seguente.



Query Access nella modalità QBE

	Camere		Prenotazioni			
Campo:	CodiceCamera		CodicePrenotazione			
Tabella:	Camere		Camere		Prenotazioni	Prenotazioni
Ordinamento:						
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteri:				No	<=[Data:]	>=[Data:]
Oppure:						

Servizio: *elenco prenotazioni clienti*

(Nome query: *ElencoPrenotazioniClienti*)

Codice sorgente SQL

```
SELECT Clienti.Cognome, Clienti.Nome, Clienti.Telefono, Clienti.Email
FROM Clienti INNER JOIN
    (NotePrenotazioni INNER JOIN Prenotazioni ON
        NotePrenotazioni.CodicePrenotazione=Prenotazioni.CodicePrenotazione)
ON Clienti.CodiceCliente = NotePrenotazioni.CodiceCliente
WHERE (Prenotazioni.Disdetta = No) AND
    (Prenotazioni.DataInizio <= [Data:]) AND
    (Prenotazioni.DataFine >= [Data:])
ORDER BY Clienti.Cognome;
```

Query Access nella modalità QBE

Campo:	Cognome	Nome	Telefono	Email	Disdetta	DataInizio	DataFine
Tabella:	Clienti	Clienti	Clienti	Clienti	Prenotazioni	Prenotazioni	Prenotazioni
Ordinamento:	Crescente						
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteri:					=No	<=[Data:]	>=[Data:]
Oppure:							

Servizio: elenco disdette clienti
 (Nome query: *ElencoDisdetteClienti*)

Codice sorgente SQL

```
SELECT Clienti.Cognome, Clienti.Nome, Clienti.Telefono, Clienti.Email
FROM Clienti INNER JOIN
    (NotePrenotazioni INNER JOIN Prenotazioni ON
        NotePrenotazioni.CodicePrenotazione=Prenotazioni.CodicePrenotazione)
ON Clienti.CodiceCliente = NotePrenotazioni.CodiceCliente
WHERE (Prenotazioni.Disdetta = Yes) AND
    (Prenotazioni.DataInizio <= [Data:]) AND
    (Prenotazioni.DataFine >= [Data:])
ORDER BY Clienti.Cognome;
```

Query Access nella modalità QBE

Campo:	Cognome	Nome	Telefono	Email	Disdetta	DataInizio	DataFine
Tabella:	Clienti	Clienti	Clienti	Clienti	Prenotazioni	Prenotazioni	Prenotazioni
Ordinamento:	Crescente						
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteri:					Si	<=[Data:]	>=[Data:]
Oppure:							

Servizio: *numero clienti presenti in un dato giorno*

(Nome query: *NumeroClientiPresentiGiorno*)

Codice sorgente SQL

```
SELECT COUNT(NotePrenotazioni.CodiceCliente) AS [Numero clienti presenti]
FROM Prenotazioni INNER JOIN NotePrenotazioni ON
    Prenotazioni.CodicePrenotazione = NotePrenotazioni.CodicePrenotazione
WHERE (Prenotazioni.Disdetta = No) AND
    (Prenotazioni.DataInizio <=[Data:]) AND
    (Prenotazioni.DataFine >= [Data:]);
```

Query Access nella modalità QBE

The screenshot shows the Access interface. At the top, a relationship diagram connects the 'Prenotazioni' table (with fields: CodicePrenotazione, CodiceCamera, TipoTrattamento, DataInizio, DataFine, Disdetta) and the 'NotePrenotazioni' table (with fields: CodicePrenotazione, CodiceCliente). A one-to-many relationship is shown between 'CodicePrenotazione' in 'Prenotazioni' and 'CodicePrenotazione' in 'NotePrenotazioni'.

Below the diagram is the Query Design view for a query named 'Numero clienti presenti: CodiceCliente'. The design grid is as follows:

Campo:	Numero clienti presenti: CodiceCliente	Disdetta	DataInizio	DataFine
Tabella:	NotePrenotazioni	Prenotazioni	Prenotazioni	Prenotazioni
Formula:	Conteggio	Dove	Dove	Dove
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteri:		No	<=[Data:]	>=[Data:]
Oppure:				