```sql
-- Registrare automaticamente le modifiche sugli immobili mantenendo lo storico modifiche
CREATE OR REPLACE FUNCTION log_modifica()
RETURNS TRIGGER AS $$
BEGIN
    -- Inserisce una nuova riga nella tabella 'modifica' per mantenere lo storico
    INSERT INTO modifica (
        id_immobile, username_utente, data_modifica, ora_modifica,
        nuovo_tipo_contratto, nuova_tipologia_immobile, nuovo_titolo, nuovo_testo,
        nuova_superficie, nuovo_prezzo, nuovo_id_indirizzo_immobile,
        nuovo_id_filtro_avanzato, nuovo_id_servizio_ulteriore
    )
    VALUES (
        OLD.id_immobile, NEW.username_agente, CURRENT_DATE, CURRENT_TIME,
        NEW.tipo_contratto, NEW.tipologia_immobile, NEW.titolo, NEW.testo,
        NEW.superficie, NEW.prezzo, NEW.id_indirizzo_immobile,
        NEW.id_filtro_avanzato, NEW.id_servizio_ulteriore
    );

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Creazione del trigger
CREATE TRIGGER trg_log_modifica
AFTER UPDATE ON immobile
FOR EACH ROW
EXECUTE FUNCTION log_modifica();
-----------------------------------------------------------------------------------------
-- Impedisce l'eliminazione di un agente se ha immobili attivi
CREATE OR REPLACE FUNCTION verifica_immobili_agente()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se l'agente ha immobili
    IF EXISTS (SELECT 1 FROM immobile WHERE username_agente = OLD.username_utente) THEN
        RAISE EXCEPTION 'Non puoi eliminare un agente con immobili attivi.';
    END IF;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_blocca_cancellazione_agente
BEFORE DELETE ON utente
FOR EACH ROW
EXECUTE FUNCTION verifica_immobili_agente();


-- Cancellazione per provare
DELETE FROM utente WHERE username_utente = 'alessandrabi';
-----------------------------------------------------------------------------------------
-- Impedisce la modifica del prezzo di un immobile se una proposta è già stata fatta su di esso
CREATE OR REPLACE FUNCTION impedisci_modifica_prezzo_con_proposta()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se esistono proposte per l'immobile
    IF EXISTS (SELECT 1 FROM proposta WHERE id_immobile_proposta = OLD.id_immobile) THEN
        RAISE EXCEPTION 'Non puoi modificare il prezzo dell''immobile perché ci sono proposte in corso.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_impendi_modifica_prezzo
BEFORE UPDATE ON immobile
FOR EACH ROW
WHEN (OLD.prezzo <> NEW.prezzo)
EXECUTE FUNCTION impedisci_modifica_prezzo_con_proposta();
-----------------------------------------------------------------------------------------
-- Non permette la cancellazione di un immobile se è stato proposto
CREATE OR REPLACE FUNCTION impedisci_cancellazione_immobile_con_proposta()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM proposta WHERE id_immobile_proposta = OLD.id_immobile) THEN
        RAISE EXCEPTION 'Non puoi cancellare un immobile che ha proposte in corso.';
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
-----------------------------------------------------------------------------------------
--  Registrazione automatica di una modifica quando un agente modifica un immobile
CREATE OR REPLACE FUNCTION registra_modifica_immobile_agente()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO modifica (
        id_immobile, username_utente, data_modifica, ora_modifica,
        nuovo_tipo_contratto, nuova_tipologia_immobile, nuovo_titolo, nuovo_testo,
        nuova_superficie, nuovo_prezzo, nuovo_id_indirizzo_immobile,
        nuovo_id_filtro_avanzato, nuovo_id_servizio_ulteriore
    )
    VALUES (
        OLD.id_immobile, NEW.username_agente, CURRENT_DATE, CURRENT_TIME,
        NEW.tipo_contratto, NEW.tipologia_immobile, NEW.titolo, NEW.testo,
        NEW.superficie, NEW.prezzo, NEW.id_indirizzo_immobile,
        NEW.id_filtro_avanzato, NEW.id_servizio_ulteriore
    );
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_registra_modifica_immobile_agente
AFTER UPDATE ON immobile
```

```
FOR EACH ROW
WHEN (NEW.username_agente IS DISTINCT FROM OLD.username_agente)
EXECUTE FUNCTION registra_modifica_immobile_agente();
-----------------------------------------------------------------------------------------------
-- Se un utente cancella il suo account viene cancellata anche le sue visite
CREATE OR REPLACE FUNCTION elimina_visite_utente()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM visita WHERE username_utente = OLD.username_utente;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_elimina_visite_utente
AFTER DELETE ON utente
FOR EACH ROW
EXECUTE FUNCTION elimina_visite_utente();
-----------------------------------------------------------------------------------------------
-- Se un utente cancella il suo account viene cancellata anche le sue proposte
CREATE OR REPLACE FUNCTION elimina_proposte_utente()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM proposta WHERE username_utente_proposta = OLD.username_utente
        OR username_agente_controproposta = OLD.username_utente;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_elimina_proposte_utente
AFTER DELETE ON utente
FOR EACH ROW
EXECUTE FUNCTION elimina_proposte_utente();
-----------------------------------------------------------------------------------------------
-- Funzione che consente l'inserimento di un immobile solo per l'agente
CREATE OR REPLACE FUNCTION verifica_ruolo_agente()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se l'utente ha il ruolo di 'Agente'
    IF (SELECT id_ruolo FROM utente WHERE username_utente = NEW.username_agente) <> 3 THEN
        RAISE EXCEPTION 'Solo un agente può inserire un immobile.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger che attiva la funzione prima dell'inserimento in immobile
CREATE TRIGGER trigger_verifica_ruolo_agente
BEFORE INSERT ON immobile
FOR EACH ROW
EXECUTE FUNCTION verifica_ruolo_agente();
-----------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION elimina_proposte_altri_stati()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se lo stato della proposta è stato cambiato in 'accettata'
    IF NEW.stato_proposta = 'Accettata' THEN
        -- Elimina tutte le altre proposte per lo stesso immobile, tranne quella appena accettata
        DELETE FROM proposta
        WHERE id_immobile_proposta = NEW.id_immobile_proposta
          AND id_proposta <> NEW.id_proposta;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_elimina_proposte_altri_stati
AFTER UPDATE ON proposta
FOR EACH ROW
WHEN (OLD.stato_proposta <> 'Accettata' AND NEW.stato_proposta = 'Accettata')
EXECUTE FUNCTION elimina_proposte_altri_stati();
-----------------------------------------------------------------------------------------------
-- Aggiorna lo stato della visita
CREATE OR REPLACE FUNCTION aggiorna_stato_visita()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.stato_approvazione_agente = 'Accettata' THEN
        NEW.stato_visita := 'Completata';
    ELSIF NEW.stato_approvazione_agente = 'Rifiutata' THEN
        NEW.stato_visita := 'Annullata';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_aggiorna_stato_visita
BEFORE UPDATE ON visita
FOR EACH ROW
WHEN (OLD.stato_approvazione_agente IS DISTINCT FROM NEW.stato_approvazione_agente)
EXECUTE FUNCTION aggiorna_stato_visita();
-----------------------------------------------------------------------------------------------
-- Trigger che aggiorna automaticamente stato_proposta a 'Controproposta' quando stato_controproposta diventa 'In attesa'.
CREATE OR REPLACE FUNCTION aggiorna_stato_proposta()
RETURNS TRIGGER AS $$
BEGIN
    -- Controlla se lo stato_controproposta è diventato 'In attesa'
    IF NEW.stato_controproposta = 'In attesa' THEN
        -- Aggiorna lo stato della proposta a 'Controproposta'
        NEW.stato_proposta := 'Controproposta';
    END IF;

    RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_aggiorna_stato_proposta
BEFORE UPDATE ON proposta
FOR EACH ROW
EXECUTE FUNCTION aggiorna_stato_proposta();
----------------------------------------
--Trigger che impedisce di effettuare una controproposta se lo stato_proposta è Accettata o Rifiutata
CREATE OR REPLACE FUNCTION public.verifica_stato_proposta()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    -- Controlla se lo stato della proposta è "Accettata" o "Rifiutata"
    IF OLD.stato_proposta = 'Accettata' OR OLD.stato_proposta = 'Rifiutata' THEN
        RAISE EXCEPTION 'Non è possibile effettuare una controproposta: la proposta è già stata %.', OLD.stato_proposta;
    END IF;

    RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.verifica_stato_proposta()
    OWNER TO postgres;
----------------------------------------
--Trigger che aggiorna automaticamente lo stato della proposta (stato_proposta) in base al nuovo valore del campo stato_controproposta.
CREATE OR REPLACE FUNCTION aggiorna_stato_proposta_da_controproposta()
RETURNS TRIGGER AS $$
BEGIN
    -- Se la controproposta è "Accettata", aggiorna lo stato della proposta
    IF NEW.stato_controproposta = 'Accettata' THEN
        NEW.stato_proposta := 'Accettata';
    ELSIF NEW.stato_controproposta = 'Rifiutata' THEN
        NEW.stato_proposta := 'Rifiutata';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_aggiorna_stato_proposta_da_controproposta
BEFORE UPDATE OF stato_controproposta ON proposta
FOR EACH ROW
EXECUTE FUNCTION aggiorna_stato_proposta_da_controproposta();
----------------------------------------
--Trigger che impedisce l'inserimento di una nuova proposta per un immobile se ne esiste già una con stato 'In attesa'.
CREATE OR REPLACE FUNCTION check_proposta_in_attesa()
RETURNS TRIGGER AS $$
DECLARE
    proposta_attesa INT;
BEGIN
    -- Controlla se esiste già una proposta in "attesa" per lo stesso immobile e per lo stesso utente
    SELECT COUNT(*) INTO proposta_attesa
    FROM proposta
    WHERE id_immobile_proposta = NEW.id_immobile_proposta
      AND username_utente_proposta = NEW.username_utente_proposta
      AND stato_proposta = 'In attesa';

    -- Se esiste una proposta in attesa per lo stesso utente, impedisce l'inserimento
    IF proposta_attesa > 0 THEN
        RAISE EXCEPTION 'Hai già una proposta in attesa per questo immobile. Attendi la conferma o il rifiuto prima di farne una nuova.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Associazione del trigger alla tabella proposta
CREATE TRIGGER trigger_check_proposta_in_attesa
BEFORE INSERT ON proposta
FOR EACH ROW
EXECUTE FUNCTION check_proposta_in_attesa();
----------------------------------------
--Questo trigger blocca l'inserimento di una nuova proposta per un immobile se esiste già una proposta accettata per lo stesso immobile
CREATE OR REPLACE FUNCTION check_proposta_accettata()
RETURNS TRIGGER AS $$
DECLARE
    proposta_accettata INT;
BEGIN
    -- Controlla se esiste già una proposta "accettata" per lo stesso immobile
    SELECT COUNT(*) INTO proposta_accettata
    FROM proposta
    WHERE id_immobile_proposta = NEW.id_immobile_proposta
      AND stato_proposta = 'Accettata';

    -- Se esiste una proposta accettata, impedisce l'inserimento e mostra il messaggio
    IF proposta_accettata > 0 THEN
        RAISE EXCEPTION 'Un''altra proposta è già stata accettata, presto l''immobile verrà eliminato.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Associazione del trigger alla tabella proposta
CREATE TRIGGER trigger_check_proposta_accettata
BEFORE INSERT ON proposta
FOR EACH ROW
EXECUTE FUNCTION check_proposta_accettata();
```